Manifold Script / .NET / XML Examples

http://www.manifold.net/

http://forum.manifold.net/Site/
http://69.17.46.171/Site/

Archive Site:
http://www.georeference.org/Default.aspx

Mapinfo-L Discussions
http://lists.directionsmag.com/discussion/list.php?f=29

Postings to GeoReference:

1. Feb 2, 2007
2.

READ ME please

For the most part, the Manifold© script examples in this document were extracted from GeoReference and Manifold-L archives or came from Manifold's "Free Stuff" Page (http://www.manifold.net/products/freestuff.html). An effort was made to provide links back to the original submissions and author information. However, the document was developed exclusively for my own training purposes and this, combined with the large size, makes it possible that ***some citations may have been missed***. **PLEASE contact me** if you feel that you were not properly referenced for your work – I will add your name to the example or cut the code from the document at your discretion and promptly update the file on GeoReference.

I have also included some of my own work. In some cases, I have clearly taken examples provided on GeoReference and modified them – if that is the case, I have tried to indicate where the idea came from and provide links back to the original work. Again, the document is large and I have probably missed some references – PLEASE contact me if you have any problems with a specific entry.

L. Ketch – Feb 2, 2007
LKetch|at|hfx[1st_dot]eastlink{2nd_dot}ca

Table of Contents

TABLE OF CONTENTS ..... **I**

V7X OBJECT MODEL (GRAPHIC) **1**

PROGRAMMING – BASICS **2**

THE VERY BASICS 2
TERMINOLOGY AND CODING CONVENTIONS – SIMPLE SCRIPT 2
REFERENCING THE ACTIVE COMPONENT WINDOW VIA SCRIPTS 3

SAMPLE CODE PROVIDED BY MANIFOLD© ON THEIR FREE STUFF PAGE **5**

AZIMUTH LINES (CREATES LINES USING AZIMUTH & DISTANCE PAIRS STORED IN TABLE) 5
BSPLINE2 (SMOOTHS SELECTED LINES) 6
BSPLINE (JOIN SELECTED POINTS WITH SPLINE) 9
Closed Spline *9*
Open Spline *11*
CENTROIDS (TRAVERSES CITIES DRAWING ADDING CENTROID OF A SET OF CITIES FOR EACH COUNTY) 13
Create Centroids *13*
Create Weighted Centroids *15*
COORDINATE SYSTEMS (REPORTS THE NUMBER OF OBJECTS AND COORDINATE SYSTEM OF EACH DRAWING INTO THE NEW COMMENTS COMPONENT) 16
Report *16*
Change Projection to Mercator *17*
COPY CREATE (COPYING FROM ONE DRAWING TO ANOTHER - CREATING A RECORD IN A TABLE) 18
Copy all objects from A to B transferring field Name *18*
Create a Record *18*
COPYOBJECTS (COPY A SELECTION TO A NEW DRAWING) 19
FORMS (DEMONSTRATES THE USE OF SEVERAL BASIC FORM CONTROLS) 20
HELLO (JSCRIPT, PEARL, PYTHON, VBSCRIPT) 21
J-Script *21*
Pearl Script *21*
Python Script *21*
VBscript *21*
IMPORT FOLDER – VBSCRIPT (IMPORTS ALL MID/MIF PAIRS IN A FOLDER AND SUB-FOLDERS) 22
NDVI (COMPUTES NORMALIZED DIFFERENCE VEGETATION INDEX – EXAMPLE OF USING THE APPLICATION STATUS BAR) 23
QUARTILES (ASSIGNS VALUES IN THE GIVEN NUMERIC COLUMN TO QUARTILES) 24
RANDOM POINTS (CREATES A DRAWING WITH POINTS RANDOMLY TAKEN FROM "DATA TABLE") 25
SEQUENTIAL LINES (JOINS POINTS SELECTED WITHIN THE ACTIVE DRAWING WINDOW WITH A LINE) 27
SPATIAL OPERATIONS (CREATE A CIRCLE AROUND EACH CITY) 28
Create a Circle Around Each City *28*
Transefer County Name to Each City *30*
Transfer Country Name to Each City using Query *32*
SURFACES (COMPUTES AVERAGE HEIGHT AND SELECTS PIXELS BELOW THIS HEIGHT) 33
Select Band *33*
Select Lower Part *34*
Select Slope *35*
TABLE GENERIC (SELECTS ALL RECORDS WITH VALUE IN AN ACTIVE COLUMN EQUAL TO THAT IN THE ACTIVE CELL) 38
Select Same Year-Month to Active *38*
Select Same to Active *38*
TABLE NWIND (ADDS A RELATION BETWEEN CATEGORIES AND PRODUCTS) 39
Add Relation between Categories and Products *39*
Clean Up Opened Table *40*
Sales by Cities Report *40*

Select Products by User Category *41*
Select Young Yet Experienced Employee *42*
TABLE NWIND FORMS (WALKS THROUGH CATEGORIES DISPLAYING LIST OF PRODUCTS FOR CURRENT CATEGORY) 44
Category Viewer *44*
Employee Viewer *46*
TEXT FORMATS (UTILITIES FOR READING DATA FROM SIMPLE ASCII FORMATS) 48
Export Released45 Text File *48*
Export Text File *49*
Import Release45 Network File *51*
Import Release45 Text File *52*
Import Text File *53*
TRANFER SELECTION (SELECT PIXELS UNDER SELECTED OBJECTS) 55
Select Pixels under Selected Objects *55*
Select Pixels with 50 Meters of Selected Objects *56*
CAPTURING SREENSHOTS OF THE ACTIVE COMPONENT 57

ACTIVE COLUMNS 63

ACTIVE COLUMNS CANNOT BE ADDED PROGRAMMATICALLY – WORK AROUND 63
CREATE ACTIVE COLUMN (SCRIPT COMPONENT MUST BE PREVIOUSLY ATTACHED TO THE TABLE) 63
CALCULATING SPHERICAL DISTANCES AND AREAS 64
TWO OR MORE ACTIVE COLUMNS IN ONE TABLE 68
CREATE A COLUMN COUNTER (SEQUENCED NUMBER) USING AN ACTIVE COLUMN 68
CALCULATE THE DISTANCE IN MILES FROM GIVEN POINT TO ALL GEOMS 71
ACTIVE COLUMN TO REFERENCE THE "PREVIOUS" RECORD IN A TABLE 72

ANALYZER OBJECT 73

SELECT TOUCH ALL OBJECTS IN ALL MAP LAYERS THAT TOUCH A BOUNDING OBJECT 73
EXPLODE POLYLINE 73
COMPUTE SHORTEST DISTANCE FROM POINT A TO A ROAD 73
ITERATING OVER THE ANALYZER OBJECTSET 74

COM OBJECT MODEL – EXTERNAL PROGRAMMING ENVIRONMENTS NET, VBA, ETC. 75

BASIC SETUP OPERATIONS TO ACCESS THE MANIFOLD.INTEROP.DLL ASSEMBLY 75
HOW TO USE THE MANIFOLD© OBJECT MODEL IN VB 2005 (PROJECT INITIAL SETUP) 77
CREATING A CUSTOM ADD-IN PANE (FROM MANIFOLD© HELP) 78
ADD-IN PANE EXAMPLE URL'S 81
MANIFOLD© COM OBJECT MODEL DESCRIPTION AND .NET USAGE (THEORY) 81
MANIFOLD OBJECT MODEL IN VB 2005 82
ADDING A COMPONENT TO AN ADDIN'S LOADED COMPONENTSET 82
CALLING EVENTARGS WITHOUT USING THE ONCLICK METHOD 84
VB.NET SAMPLES 85
LOADING AND COMPILING THE MAPVIEWER PROJECT – CONVERTING TO VB2005 86
7X COM/REGISTRY PROBLEM? 87
PROGRESSBAR OBJECT 88
USING THE VBA PROGRAMMING ENVIRONMENT 88
CREATE A MAP PROJECT FILE AND DRAWING THROUGH COM OR .NET 95
CREATING A VIEW WITH A SPECIFIC SCALE 96
CANCELLING THE PROGRESSBAR OBJECT FROM VB.NET 98
PROGRAMMING AN ADD-IN TOOLBAR DLL IN VB.NET 99
HOW TO CREATE A COMPONENT VIEW AT THE DESIRED SCALE VIA VB.NET 99
BUILD A FORM VIA VB.NET CODE 101

COORDINATE SYSTEM MANIPULATION 103

SETTING CURRENT PROJECTION OF IMAGES 103
APPLY A NEW PROJECTION TO AN EXISTING DRAWING 104
CHANGE PROJECTIONS TO LAT-LONG (NO OFFSETS) 105
XL VBA – CREATE A COMPONENTS PROJECTION REPORT 106

Reset Orthographic Projection Centre to Map Window Centre 109
Reassign Component Coordinate System & Datum - keep Original Scales & Offsets 109
Applying a Lat/Long Coordinate System to a Point 111
Improving Performance when using CoordinateConverter.Convert() 113
Convert Coordinate System via Code 115
Tweaking a Surface Coordinate System 117
Verifying the Coordinate System for a Large Number of Components 118
Batch Reproject Drawings 119
Build Coordinate System Object from WKT String 120
Convert a Point from Drawing Coordinate System to Lat/Lon 121
Extract Map Layer Coordinate System Information to Table Columns 122

IMAGES, SURFACES AND TERRAINS 127

Terrain Fly-Through 127
Terrain Fly-Through: Save the X, Y and Z Flight Coordinates 133
Surface Analysis Script – slope, orientation, vertical & horizontal curvature calculation 135
Extracting a Surface Height at a given Latitude/Longitude 140
Batch Transform Image Gamma 142
Sum all Surfaces in a Project 143
Extract the Surface Height for Each Point in Drawing 143

CALCULATING HILLSHADE FROM A DEM 145

IMPORT / EXPORT 146

Import Surfaces, Set Surface Projections, then Merge to One Surface 146
Import External Images Using Drawing Index 153
Export (to Manifold Projects) Internal Drawing Tiles Using Drawing Index 154
Export a Surface Straight to Image Format 156
Exporting Surfaces as Images 156
Exporting All Images in a Project 157
Exporting Geometry as SHP MetaData 157
Export Surface Pixels Greater than 0 to XYZ Text File (FileSystemObject) 165
Generic Exporter 166
Generate and Export Tile Images 168
Export Table to MS EXCEL 170
Export to KML 171

IMS 173

IMS Selections from Queries not being Updated/Refreshed 173
Manipulating IMS Layers via Code 174
Layer Code (turn on/off) IMS 175
How to have Some Layers Unchecked when a Webpage is Loaded? 176
Set Layer to Layer Off 177
Published Layers 179
How to Disable Layers on Startup 181

LABELS 185

Adding Labels via Code 185
Creating Bound Labels 185
Bound Labels Created by Script do Not Display 186
Script to Place Labels on Point Objects 188
Adding Labels at x/y Points to an Unbound Label Component 189
Set Label Rotation Based upon a Column Value 190

LAYOUTS 191

Applying a Layout-Template to Different Layouts 191
Automate Layout Creation 191
Custom Layout Templates 194

LINES **195**

LINE INTERSECTS LINE AT COORDINATES? 195
FIND LINE INTERSECT COORDINATES 195
PLOTTING POINTS EQUIDISTANT ALONG A LINE - SCRIPT 199
Approach 1 ..... *199*
Approach 2 ..... *202*
FIND COORDINATES OF INFLECTION POINTS ALONG A LINE 203
CREATING TRACK WITH DIFFERENT POINT FORMAT FOR START AND END (INTERESTING CODE FOR MAKING LINES, ETC.) 204

MAP COMPONENTS **208**

FIND A MAP WINDOW EXTENTS, CENTER AND SCALE 208
SELECT TOUCH ALL OBJECTS IN ALL MAP LAYERS THAT TOUCH A BOUNDING OBJECT 208
THEMATICALLY MAPPING PRECIPITATION AMOUNTS USING POINT THEMES 209
LISTING THE PROJECTION FOR EACH MAP LAYER 211
CHECKING THE WINDOWSET FOR A MAPCOMPONENT THEN FINDING THE ACTIVE DRAWING 215
ZOOM TO SELECTED OBJECT IN MAP WINDOW AND SAVE AN IMAGE 216
SETTING MAP LAYER ORDER BY CODE 217
TURNING LAYERS ON AND OFF VIA A SCRIPT 217
COLLECT A MAP WINDOW PARAMETER SET (COORDSYS, WIDTH, CENTERX, ETC) 217

MISCELLANEOUS **219**

MANIFOLD SHOULD NEVER CRASH 219
YOU CAN NOT CURRENTLY CONTOUR DATA USING A SCRIPT 219
CALCULATING SPHERICAL DISTANCES AND AREAS IN MANIFOLD© — (HURDAT) 219
CREATE A COMPONENT AT THE ROOT LEVEL (NOT IN AN EXISTING FOLDER) 223
COPY A COMPONENT FROM ANOTHER MAP FILE 223
APPLYING THEMATIC LEGENDS 224
CALLING ONE SCRIPT FROM ANOTHER SCRIPT 224
PROPER WAY TO ASSIGN AN OBJECT TO A COMPONENT ADDED BY THE .NEW METHOD 225
ADDING A DLL REFERENCE TO A VBSCRIPT 225
IMPLEMENTING "DISSOLVE" 226
USE "UNION RATHER THEN "DISSOLVE" IN VB SCRIPT 227
CHANGING DRAWING AREA COLOURS AND THEMES 229
CONVERT TABLE QUERY TO DELIMITED TEXT STORED IN COMMENTS COMPONENT 230
SEARCH ALL SCRIPTS FOR A SPECIFIC STRING 235
SET DEFAULT POINT SIZE AND STYLE 235
QUERY CACHING – BE CAREFUL WHEN RERUNNING THE SAME QUERY.TEXT VIA CODE. 236
CANNOT LINK A QUERY PROGRAMMATICALLY 237
ZOOM TO SELECTED AREAS (WITH ADDITIONAL BORDER) 237
CREATE A "ZOOMTO" RECT OBJECT SIZED A PERCENTAGE LARGER THAN THE ORIGINAL DRAWING COMPONENT RECT 239
DOES A COMPONENT NAME EXIST IN THE PROJECT? 241
DELETE A PROJECT COMPONENT (PASS COMPONENT NAME) 241
"PASTE AS" COMPONENT TYPE (RETURNS NEW COMPONENT INDEX) 242
DUPLICATE A COMPONENT 244
CLOSE A WINDOW 249
RETRIEVE ALL COMPONENT NAMES 249
RETRIEVE OPEN WINDOW INDEXES AND NAMES (INTO DYNAMIC ARRAY) 250
CLOSE A WINDOW 252
UNSELECT ALL PROJECT COMPONENTS (TYPICALLY USED IN MAPS) 252
IMPORT COMPONENTS FROM OTHER PROJECTS 253
MERGE GEOBASE DEM TILES 253
USING REGULAR EXPRESSIONS 256
GET THE LAST TOKEN IN A STRING (IN PROPER CASE) 258
BUBBLESORT TEXT (BASED ON UPPER CASE TEXT EVALUATION) 259
FIND THE NUMBER OF DAYS IN ANY MONTH (INCLUDING FEB LEAP YEARS) 259

Is Year a Leap-Year 260
Using SendKeys to Load Dialogs 261
You Cannot Instantiate the Root of the WSH Object Model (WScript) 263
Casting Geom (I) – Binary Column Data Not Equal a GEOM Object 264
NewImport Problems – External Database Connection Fails 265
Export a Palette using a Script 265

PRESET DEFININTIONS: COMMAND FILTER LIST 267

PRESET DEFININTIONS: DISTANCE AND AREA UNITS 276

PRESET DEFINITIONS: STYLE LISTS (SYSTEM DEPENDANT) 278

Table: V7 Area styles (Aug 12, 2006): 278
Table: V7 Line styles (Aug 12, 2006): 278
Table: V7 Point styles (Aug 12, 2006): 279
Table: Label styles (Aug 12, 2006): 282

OBJECT MODEL – V7 FEATURES AND CHANGES 283

Deselect Records – New V7 SelectNone, SelectAll & SelectInverse Methods 283
"Document" is a Predefined Keyword – ActiveDocument Often not Required 283
Accessing the Records.LastAdded Object 284
Dumping V7X Formatting Options – Style Lists 285

OBJECTS 286

IDs Assigned to New Drawing Objects are not Guaranteed to be Sequential 286
Adding/Editing Objects in a Drawing Layer 286
How to Add Objects (Points) to a Drawing 287
Creating a New Geom and Adding it to a Drawing ObjectSet 287
Extract Object Coordinates via Script Driven SQL 289
Extract Object Coordinates via Object Model 292

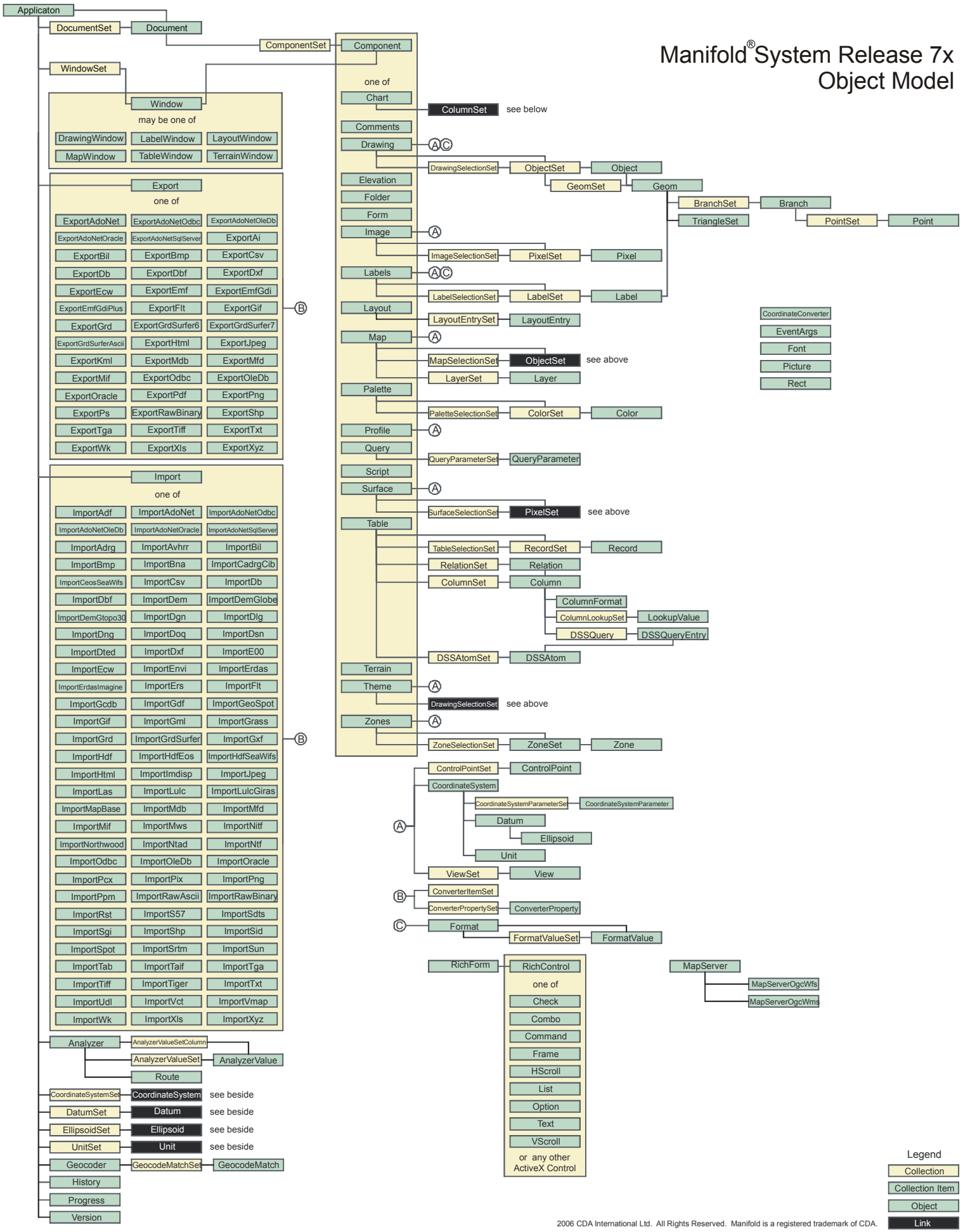
TABLES 299

Add Columns to a Table 299
Adding Columns to a Table via Script / Manifold© COLUMN ENUMERATIONS 300
Scripting Table Relationships 301
Access Column Data by Index Rather than Column Name 302
Finding and Deleting Duplicate Records based upon Column Contents 303
Save Drawing Names to a Column 304
Return the Identity Column Name 307
Collect Collumn Names (All, Regular, Intrinsics only, etc) 308
Trim all Text Columns in a Table 311
Create a Drawing from a Table that Contains a Manifold Geometry Column 312

XML – CUSTOM STYLES, ETC 317

Defining Custom Label Styles (Points and Areas can be done as well) 317
Line: River Bank 318
Point: Hourglass 318
Example Equal-Interval Theme 319
Adding a True Type Font Symbol 320
XML Schema CANNOT be used to validate a File 321
Rotating Symbols on a Continuous Scale 323
Custom Layout Templates 323
VB.NET Reader for XML Units File 324

V7x Object Model (Graphic)



2006 CDA International Ltd. All Rights Reserved. Manifold is a registered trademark of CDA.

Programming – Basics

A topic to introduce basic concepts and terminology used to program using Manifold scripting. http://www.georeference.org/DNN31/KnowledgeBase/Articles/ProgrammingBasics/tabid/86/Default.aspx

Manifold provides scripting support via Microsoft's ActiveX scripting environment from with the Manifold program and via a Manifold System Type Library from other programming environments. This topic will concentrate on the scripting environment.

The Very Basics

Programming basic terminology is assumed. If you are not at all familiar with programming obtain an entry level text on that use VBScript or VBA as the introductory language.

In a "learn by doing" methodology start Manifold and import a simple drawing – name it 'Drawing' for use with this tech topic.

Create a script object via the Project Window and open it.

```
Sub Main
   Application.MessageBox "Hello, World!"
End Sub
```

Click the Run or Run Under Debugger icon on the main toolbar  and you will see the Hello World! Message.

Terminology and Coding Conventions – Simple Script

In the simple example bellow I will use terms in context as a way to define the terms. User defined variables will begin with lower case letters and Manifold objects will begin with upper case. VBScript reserve words will also begin with upper case.

To get started with a typical script you will what to get a reference to component that is showing in your Project Pane. In this example case the drawing component named 'Drawing'.

Change your script to read:

```
Sub Main
    Set document = Application.ActiveDocument                                    '1
    Set components = document.ComponentSet                                       '2
    Application.MessageBox "There are " & components.Count & " components in the
document." '3
    Set component = components("Drawing")                                        '4
    Application.MessageBox "Component: " & component.TypeName & ":" & _
         component.Name, ,"Hello World!"                                         '5
End Sub
```

The first line stores a reference to the project file you have opened. Application.ActiveDocument returns a reference to an object of the type Document and the Set VBScript function stores it in the variable document. The Application object is a bit different from most other objects in that it is always present and does not need a Set statement in order to be used. Think of the Application object as representing the running instance on the Manifold program.

Objects have Properties, single attribute values and collections, and provides Methods to manipulate the object. In the second line document is an object and its collection of Components, ComponentSet is being assign to the variable components.

Collections are multiple instances of objects. Objects within collections can be accessed via an index number and often via name. The third line displays the number of components in your project by using the Count property of the ComponentSet collection that was stored in the variable components.

The forth line extracts a reference to the drawing component we want to work with by using its name.

The last line displays the component's type and name by using the Components TypeName and Name properties.

Referencing the Active Component Window via Scripts

Getting access to a component by its name is interesting but in practice you don't know the name of the drawing(s) you will be working with. More likely you will want the script to operate upon the component the "has the focus" in the main Manifold window – obtained via the ActiveWindow property in scripts. (The component with the focus will have a title bar that is brighter than the rest.) If the active window is a map you'd like the script to operate on the active layer within the map.

Here is some code to get a reference to the component that has the focus. This script will also check to see if the component is of the 'correct' type for this script and complains if it isn't.

```
Sub Main
    Set document = Application.ActiveDocument
    Set components = document.ComponentSet
    Set windows = Application.WindowSet

    ' -- fail if there are no opened windows
    If windows.Count = 0 Then
        Application.MessageBox "No active window"
        Exit Sub
    End If

    Set window = windows.ActiveWindow
    Set component = window.Component

    If component.Type = ComponentMap Then ' drill down to active layer
```

```
        Set component = window.ActiveComponent
    End If

    ' real work starts here
    ' test for correct document type
    If component.Type <> ComponentDrawing Then
        Application.MessageBox "Active window is not a Drawing Component"
        Exit Sub
    End If

    Application.MessageBox component.TypeName & ": " &
component.Name,,”ActiveWindow”
End Sub
```

Sample Code Provided by Manifold© on Their Free Stuff Page

http://www.manifold.net/products/freestuff.html

Azimuth Lines (creates lines using azimuth & distance pairs stored in table)

```
'Create Lines by Azimuth and Distance Pairs

'Creates lines using set of azimuth-and-distance pairs stored
'in table
'[Lines].  Table structure:

'  Name - name used to uniquely identify the line,
'  Seq - sequential number of record within the line,
'  X - X of starting location or azimuth to next location,
'  Y - Y of starting location or distance to next location.

Sub Main
   Set components = document.ComponentSet

   ' -- locate and execute query
   queryIndex = components.ItemByName("Lines Query")
   If queryIndex < 0 Then
      Application.MessageBox "'Lines Query' can not be found"
      Exit Sub
   End If
   Set query = components(queryIndex)
   Set queryTable = query.Table

   ' -- create new drawing
   Set drawing = document.NewDrawing("Drawing")

   ' -- obtain drawing table and equip it with 'Name' column
   Set drawingTable = drawing.OwnedTable
   Set drawingName = drawingTable.ColumnSet.NewColumn
   drawingName.Name = "Name"
   drawingName.Type = ColumnTypeWText
   drawingTable.ColumnSet.Add(drawingName)

   pi = 3.141592653589793

   ' -- scan query adding lines as necessary
   Set records = queryTable.RecordSet
   recordIndex = 0
   recordCount = records.Count
   Do While recordIndex < recordCount

      Set record = records(recordIndex)

      ' -- obtain name of creating line
      name = CStr(record.Data("Name"))

      Set pointSet = Application.NewPointSet

      ' -- log starting location
```

```
      Set point = Application.NewPoint
      point.X = CDbl(record.Data("X"))
      point.Y = CDbl(record.Data("Y"))
      pointSet.Add(point)

      ' -- advance to next record
      recordIndex = recordIndex + 1

      ' -- create line
      Do While recordIndex < recordCount
         Set record = records(recordIndex)

         ' -- check for start of next line
         nameCurrent = CStr(record.Data("Name"))
         If LCase(name) <> LCase(nameCurrent) Then
            Exit Do
         End If

         azimuth = CDbl(record.Data("X")) * pi / 180
         distance = CDbl(record.Data("Y"))
         x = point.X + distance*Sin(azimuth)
         y = point.Y + distance*Cos(azimuth)

         ' -- log current location
         Set point = Application.NewPoint
         point.X = x
         point.Y = y
         pointSet.Add(point)

         ' -- advance to next record
         recordIndex = recordIndex + 1
      Loop

      ' -- append created line if its metric has at least two locations
      If pointSet.Count > 1 Then
         Set geom = Application.NewGeom(GeomLine, pointSet)

         ' -- append line
         drawing.ObjectSet.Add(geom)

         ' -- supply line name to autogenerated line record
         ' -- WARNING: it would be safer to obtain newly created object
         ' --   with drawing.LastAdded and locate record using its ID property.
         drawingTable.RecordSet(drawingTable.RecordSet.Count - 1).Data("Name") = name
      End If
   Loop

   drawingTable.Open
   drawing.Open
End Sub
```

BSpline2 (smooths selected lines)

```
'Splinearize Selected Lines
```

```
'Smooths lines selected within the active drawing window
'by 3-splines.

' Smooths lines selected within the active drawing window by 3-splines.

Sub Main
   Set windows = Application.WindowSet

   ' fail if there are no opened windows
   If windows.Count = 0 Then
      Application.MessageBox "No active window"
      Exit Sub
   End If

   Set window = windows.ActiveWindow
   Set component = window.Component

   ' fail if active component is not a drawing
   If component.Type <> ComponentDrawing Then
      Application.MessageBox "Active component is not a drawing"
      Exit Sub
   End If

   ' process each selected line
   Set objects = component.Selection
   For Each object In objects
      If object.Type = ObjectLine Then
         SmoothLine(object)
      End If
   Next
End Sub

' ****************************************************************************

Sub SmoothLine(object)
   Set branches = object.Geom.BranchSet

   ' reject objects with more than one branch (use Decompose transform to split such
objects into parts)
   If branches.Count <> 1 Then
      Exit Sub
   End If

   ' reject objects with less than three points
   Set pointsOrg = branches(0).PointSet
   If pointsOrg.Count <= 2 Then
      Exit Sub
   End If

   ' compose intermediate point set from original point set
   ' duplicating first and last points
   Set pointsInt = Application.NewPointSet
   Set point = Application.NewPoint
   point.X = pointsOrg(1).X
   point.Y = pointsOrg(1).Y
   pointsInt.Add point
```

```
For index = 0 to pointsOrg.Count-1
   Set point = Application.NewPoint
   point.X = pointsOrg(index).X
   point.Y = pointsOrg(index).Y
   pointsInt.Add point
Next
Set point = Application.NewPoint
point.X = pointsOrg(pointsOrg.Count-2).X
point.Y = pointsOrg(pointsOrg.Count-2).Y
pointsInt.Add point

' compose resulting point set interpolating the intermediate point set
Set pointsFin = Application.NewPointSet
Set point = Application.NewPoint
point.X = pointsInt(1).X
point.Y = pointsInt(1).Y
pointsFin.Add point
For index = 1 to pointsOrg.Count - 1
   Set pointA = pointsInt(index-1)
   Set pointB = pointsInt(index)
   Set pointC = pointsInt(index+1)
   Set pointD = pointsInt(index+2)

   dx = pointB.X
   dy = pointB.Y
   cx = pointB.X - pointA.X
   cy = pointB.Y - pointA.Y
   If index = 1 Then
      cx = -cx
      cy = -cy
   End if
   cx = (cx + pointC.X - pointB.X) / 2
   cy = (cy + pointC.Y - pointB.Y) / 2

   bx = pointD.X - pointC.X
   by = pointD.Y - pointC.Y
   If index = (pointsOrg.Count - 1) Then
      bx = -bx
      by = -by
   End if
   bx = (bx + pointC.X - pointB.X) / 2
   by = (by + pointC.Y - pointB.Y) / 2
   bx = 3 * pointC.X - 2 * cx - 3 * dx - bx
   by = 3 * pointC.Y - 2 * cy - 3 * dy - by

   ax = pointC.X - bx - cx - dx
   ay = pointC.Y - by - cy - dy

   For i = 0 to 19
      t = i / 20.0
      Set point = Application.NewPoint
      point.X = ax * t * t * t + bx * t * t + cx * t + dx
      point.Y = ay * t * t * t + by * t * t + cy * t + dy
      pointsFin.Add point
   Next
Next
Set point = Application.NewPoint
```

```
   point.X = pointsInt(pointsOrg.Count).X
   point.Y = pointsInt(pointsOrg.Count).Y
   pointsFin.Add point

   ' add new branch and remove the former branch
   branches.Add pointsFin
   branches.Remove 0
End Sub
```

BSpline (join selected points with spline)

```
'Join Selected Points with Spline

'Joins points selected within the active drawing window with
'an opened 3-spline.  The order of points on a spline is
'determined by their IDs.

'Join Selected Points with Closed Spline

'Joins points selected within the active drawing window with
'a closed b-spline.  The order of points on a spline is
'determined by their IDs.
```

Closed Spline

```
' Joins points selected within the active drawing window with
' a closed b-spline.

Sub Main
   Set windows = Application.WindowSet

   ' fail if there are no opened windows
   If windows.Count = 0 Then
      Application.MessageBox "No active window"
      Exit Sub
   End If

   Set window = windows.ActiveWindow
   Set component = window.Component

   ' fail if active component is not a drawing
   If component.Type <> ComponentDrawing Then
      Application.MessageBox "Active component is not a drawing"
      Exit Sub
   End If

   ' create temporary query component
   Set components = document.ComponentSet
   Set query = document.NewQuery("Temp")
   query.Text = "SELECT ID, [X (I)], [Y (I)] FROM [" + component.Name + "] WHERE
[Selection (I)] AND [Type (I)] = 1 ORDER BY ID;"
```

```
   ' KLUDGE: verbatim value of 1 in '[Type (I)] = 1' fragment filters out line and
area objects
   Set table = query.Table

   ' ensure there are at least two records
   Set records = table.RecordSet
   If records.Count < 3 Then

      ' remove temporary query component
      components.Remove(components.ItemByID(query.ID))

      Application.MessageBox "There should be at least three selected points"
      Exit Sub
   End If

   Set temppointSet = Application.NewPointSet
   For recordIndex = 0 to records.Count-1
      Set record = records(recordIndex)
      Set point = Application.NewPoint
      point.X = CDbl(record.Data("X (I)"))
      point.Y = CDbl(record.Data("Y (I)"))
      tempPointSet.Add point
   Next
   For recordIndex = 0 to 2
      Set record = records(recordIndex)
      Set point = Application.NewPoint
      point.X = CDbl(record.Data("X (I)"))
      point.Y = CDbl(record.Data("Y (I)"))
      tempPointSet.Add point
   Next

   ' create new point set object
   Set pointSet = Application.NewPointSet

   ' scan queried table adding locations to point set
   For recordIndex = 0 to records.Count - 1
      For i = 0 to 19
         t = i / 20.0
         C1 = (1 - t) * (1 - t) * (1 - t) / 6
         C2 = (3 * t * t * t - 6 * t * t + 4) / 6
         C3 = (-3 * t * t * t + 3 * t * t + 3 * t + 1) / 6
         C4 = t * t * t / 6
         Set point = Application.NewPoint
         point.X =  C1 * tempPointSet.Item(recordIndex).X + C2 *
tempPointSet.Item(recordIndex + 1).X + C3 * tempPointSet.Item(recordIndex + 2).X + C4
* tempPointSet.Item(recordIndex + 3).X
         point.Y =  C1 * tempPointSet.Item(recordIndex).Y + C2 *
tempPointSet.Item(recordIndex + 1).Y + C3 * tempPointSet.Item(recordIndex + 2).Y + C4
* tempPointSet.Item(recordIndex + 3).Y
         PointSet.Add point
      Next
   Next
   Set point = Application.NewPoint
   point.X =  PointSet.Item(0).X
   point.Y =  PointSet.Item(0).Y
   PointSet.Add point
```

```
   ' create new geometric entity
   Set geom = Application.NewGeom(GeomLine, pointSet)

   ' create new line object
   component.ObjectSet.Add geom

   ' remove temporary query component
   components.Remove(components.ItemByID(query.ID))
End Sub
```

Open Spline

```
' Joins points selected within the active drawing window with an
' opened 3-spline.

Sub Main
   Set windows = Application.WindowSet

   ' fail if there are no opened windows
   If windows.Count = 0 Then
      Application.MessageBox "No active window"
      Exit Sub
   End If

   Set window = windows.ActiveWindow
   Set component = window.Component

   ' fail if active component is not a drawing
   If component.Type <> ComponentDrawing Then
      Application.MessageBox "Active component is not a drawing"
      Exit Sub
   End If

   ' create temporary query component
   Set components = document.ComponentSet
   Set query = document.NewQuery("Temp")
   query.Text = "SELECT ID, [X (I)], [Y (I)] FROM [" + component.Name + "] WHERE
[Selection (I)] AND [Type (I)] = 1 ORDER BY ID;"
   ' KLUDGE: verbatim value of 1 in '[Type (I)] = 1' fragment filters out line and
area objects
   Set table = query.Table

   ' ensure there are at least two records
   Set records = table.RecordSet
   If records.Count < 2 Then

      ' remove temporary query component
      components.Remove(components.ItemByID(query.ID))

      Application.MessageBox "There should be at least two selected points"
      Exit Sub
   End If

   Set temppointSet = Application.NewPointSet
```

```
   Set record = records(1)
   Set point = Application.NewPoint
   point.X = CDbl(record.Data("X (I)"))
   point.Y = CDbl(record.Data("Y (I)"))
   tempPointSet.Add point
   For recordIndex = 0 to records.Count-1
      Set record = records(recordIndex)
      Set point = Application.NewPoint
      point.X = CDbl(record.Data("X (I)"))
      point.Y = CDbl(record.Data("Y (I)"))
      tempPointSet.Add point
   Next
   Set record = records(records.Count-2)
   Set point = Application.NewPoint
   point.X = CDbl(record.Data("X (I)"))
   point.Y = CDbl(record.Data("Y (I)"))
   tempPointSet.Add point

   ' create new point set object
   Set pointSet = Application.NewPointSet

   ' scan queried table adding locations to point set
   For recordIndex = 1 to records.Count - 1
      dx = tempPointSet.Item(recordIndex).X
      dy = tempPointSet.Item(recordIndex).Y

      cx = tempPointSet.Item(recordIndex).X - tempPointSet.Item(recordIndex - 1).X
      cy = tempPointSet.Item(recordIndex).Y - tempPointSet.Item(recordIndex - 1).Y
      If recordIndex = 1 Then
         cx = -cx
         cy = -cy
      End if
      cx = (cx + tempPointSet.Item(recordIndex + 1).X -
tempPointSet.Item(recordIndex).X) / 2
      cy = (cy + tempPointSet.Item(recordIndex + 1).Y -
tempPointSet.Item(recordIndex).Y) / 2

      bx = tempPointSet.Item(recordIndex + 2).X - tempPointSet.Item(recordIndex + 1).X
      by = tempPointSet.Item(recordIndex + 2).Y - tempPointSet.Item(recordIndex + 1).Y
      If recordIndex = (records.Count - 1) Then
         bx = -bx
         by = -by
      End if
      bx = (bx + tempPointSet.Item(recordIndex + 1).X -
tempPointSet.Item(recordIndex).X) / 2
      by = (by + tempPointSet.Item(recordIndex + 1).Y -
tempPointSet.Item(recordIndex).Y) / 2
      bx = 3 * tempPointSet.Item(recordIndex + 1).X - 2 * cx - 3 * dx - bx
      by = 3 * tempPointSet.Item(recordIndex + 1).Y - 2 * cy - 3 * dy - by

      ax = tempPointSet.Item(recordIndex + 1).X - bx - cx - dx
      ay = tempPointSet.Item(recordIndex + 1).Y - by - cy - dy

      For i = 0 to 19
         t = i / 20.0
         Set point = Application.NewPoint
         point.X =  ax * t * t * t + bx * t * t + cx * t + dx
```

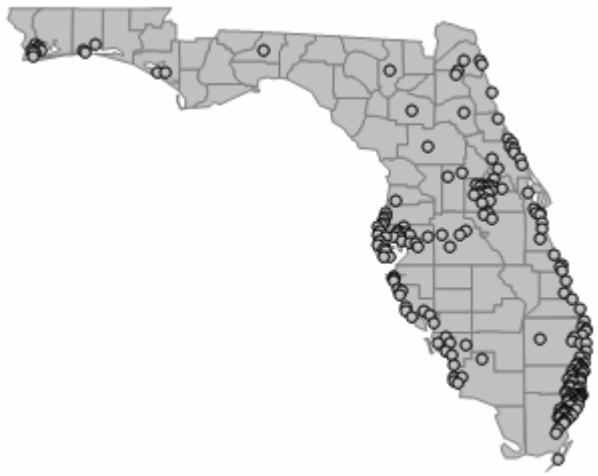
```
            point.Y =  ay * t * t * t + by * t * t + cy * t + dy
            PointSet.Add point
         Next
      Next
      Set point = Application.NewPoint
      point.X =  tempPointSet.Item(records.Count).X
      point.Y =  tempPointSet.Item(records.Count).Y
      PointSet.Add point

      ' create new geometric entity
      Set geom = Application.NewGeom(GeomLine, pointSet)

      ' create new line object
      component.ObjectSet.Add geom

      ' remove temporary query component
      components.Remove(components.ItemByID(query.ID))
End Sub
```

Centroids (traverses Cities drawing adding centroid of a set of cities for each county)



Create Centroids

```
' Traverses Cities drawing adding centroid of a set of cities for each county.

Sub Main
   Set components = document.ComponentSet

   ' -- locate target drawing
   Set drawing = components(components.ItemByName("Cities"))

   ' -- locate and execute helper query
   Set query = components(components.ItemByName("Cities by County"))
   Set table = query.Table
   Set records = table.RecordSet
```

```
   county = ""
   cities = 0
   x = 0
   y = 0

   Application.StatusText = "Scanning table"

   ' -- scan queried table creating centroids as necessary
   For recordIndex = 0 To records.Count-1
      Application.StatusText = "Scanning table (" & CStr(recordIndex+1) & " of " &
CStr(records.Count) & ")"

      Set record = records(recordIndex)

      ' -- acquire record data
      countyNew = CStr(record.Data("County"))
      xNew = CDbl(record.Data("X (I)"))
      yNew = CDbl(record.Data("Y (I)"))

      If countyNew <> county Or recordIndex = records.Count-1 Then
         If cities > 0 Then

            ' -- create centroid
            Set pointSet = Application.NewPointSet
            Set point = Application.NewPoint
            point.X = x / cities
            point.Y = y / cities
            pointSet.Add point
            Set geom = Application.NewGeom(GeomPoint, pointSet)

            Set objectSet = drawing.ObjectSet

            ' -- append centroid to drawing
            objectSet.Add geom

            ' -- select newly created centroid
            objectSet.LastAdded.Mask = 1
         End If

         cities = 1
         county = countyNew
         x = xNew
         y = yNew
      Else
         cities = cities + 1
         x = x + xNew
         y = y + yNew
      End If
   Next

   Application.StatusText = "Done"
End Sub
```

Create Weighted Centroids

```
'Traverses Cities drawing adding weighted centroid of a set of cities for
'each county.  Weight of city is taken from its population field.

Sub Main
   Set components = document.ComponentSet

   ' -- locate target drawing
   Set drawing = components(components.ItemByName("Cities"))

   ' -- locate and execute helper query
   Set query = components(components.ItemByName("Cities by County"))
   Set table = query.Table
   Set records = table.RecordSet

   county = ""
   weight = 0
   x = 0
   y = 0

   Application.StatusText = "Scanning table"

   ' -- scan queried table creating centroids as necessary
   For recordIndex = 0 To records.Count-1
      Application.StatusText = "Scanning table (" & CStr(recordIndex+1) & " of " &
CStr(records.Count) & ")"

      Set record = records(recordIndex)

      ' -- acquire record data
      countyNew = CStr(record.Data("County"))
      weightNew = CDbl(record.Data("Population"))
      xNew = CDbl(record.Data("X (I)"))
      yNew = CDbl(record.Data("Y (I)"))

      ' -- protect from invalid weights
      If weightNew <= 0 Then
         weightNew = 1
      End If

      If countyNew <> county Or recordIndex = records.Count-1 Then
         If weight > 0 Then

            ' -- create centroid
            Set pointSet = Application.NewPointSet
            Set point = Application.NewPoint
            point.X = x / weight
            point.Y = y / weight
            pointSet.Add point
            Set geom = Application.NewGeom(GeomPoint, pointSet)

            Set objectSet = drawing.ObjectSet

            ' -- append centroid to drawing
            objectSet.Add geom
```

```
            ' -- select newly created centroid
            objectSet.LastAdded.Mask = 1
         End If

         county = countyNew
         weight = weightNew
         x = xNew*weightNew
         y = yNew*weightNew
      Else
         weight = weight + weightNew
         x = x + xNew*weightNew
         y = y + yNew*weightNew
      End If
   Next

   Application.StatusText = "Done"
End Sub
```

Coordinate Systems (reports the number of objects and coordinate system of each drawing into the new Comments component)

Report

```
'Reports number of objects and coordinate system of each drawing into
'the new Comments component.

Sub Main
   Set Components = Document.ComponentSet

   ' -- create report component or reuse existing one
   ReportIndex = Components.ItemByName("Drawing Report Text")
   If ReportIndex < 0 Then
      Set Report = Document.NewComments("Drawing Report Text")
   Else
      Set Report = Components(ReportIndex)
      If  Report.Type <> ComponentComments Then
         Application.MessageBox "Comments component required"
         Exit Sub
      End If

      ' -- reset component text
      Report.Text = ""
   End If

   ' -- append report header
   Report.AddText "Report" & vbCrLf
   Report.AddText "------" & vbCrLf
   Report.AddText vbCrLf
   Report.AddText "Date: " & CStr(Now)
   Report.AddText vbCrLf
   Report.AddText vbCrLf
```

```
   ' -- traverse project components
   For Index = 0 To Components.Count-1
      Set Component = Components(Index)
      If  Component.TypeName = "Drawing" Then
         Set Objects = Component.ObjectSet
         Set System = Component.CoordinateSystem

         ' -- append drawing to report
         Report.AddText Component.Name & vbCrLf
         Report.AddText "  number of objects: " & CStr(Objects.Count) & vbCrLf
         Report.AddText "  coordinate system: " & System.Preset & vbCrLf
         Report.AddText "  coordinate system base: " & System.Name & vbCrLf
      End If
   Next

   Report.Open
End Sub
```

Change Projection to Mercator

```
'Projects [European Cities] to customized version of the Mercator projection.

Sub Main

   ' -- locate 'European Cities'
   Set Components = Application.ActiveDocument.ComponentSet
   DrawingIndex = Components.ItemByName("European Cities")
   If DrawingIndex < 0 Then
      Application.MessageBox "'European Cities' drawing not found."
      Exit Sub
   End If
   Set Drawing = Components(DrawingIndex)

   ' -- create target coordinate system
   Set Target = Application.NewCoordinateSystem("Mercator")
   Target.Datum = Application.NewDatum("Clarke 1866")
   Target.Unit = Application.NewUnit("Foot")

   ' -- create coordinate system converter
   Set Converter = Application.NewCoordinateConverter

   ' -- prepare coordinate converter
   converter.Prepare Drawing.CoordinateSystem, Target

   Set Objects = Drawing.ObjectSet

   ' -- project all objects within drawing
   For IndexObj = 0 To Objects.Count-1
      Set Object = Objects(IndexObj)
      converter.Convert Object.Geom
   Next

   ' -- modify coordinate system within drawing
   Drawing.CoordinateSystem = Target
```

```
   Drawing.Open
End Sub
```

Copy Create (copying from one drawing to another - creating a record in a table)

```
' These scripts provide simple examples of copying from one drawing to another,
' as well as creating a record in a table.
```

Copy all objects from A to B transferring field Name

```
Sub Main
   Set components = document.ComponentSet
   Set drawingA = components(components.ItemByName("A"))
   Set drawingB = components(components.ItemByName("B"))
   Set objectsB = drawingB.ObjectSet ' instantiate to use LastAdded property!
   Set recordsA = drawingA.OwnedTable.RecordSet
   Set recordsB = drawingB.OwnedTable.RecordSet

   For index = 0 To drawingA.ObjectSet.Count-1
      Set object = drawingA.ObjectSet(index)

      ' add object
      objectsB.Add object.Geom

      Set recordA = recordsA(recordsA.ItemByID(object.ID))
      Set recordB = recordsB(recordsB.ItemByID(objectsB.LastAdded.ID))

      ' add object name
      recordB.Data("Name") = recordA.Data("Name")
   Next
End Sub
```

Create a Record

```
' Create new record in table C and fill it with data.

Sub Main
   Set components = document.ComponentSet
   Set table = components(components.ItemByName("C"))
   Set records = table.RecordSet ' instantiate to use LastAdded property!

   ' add record
   records.AddNew

   Set record = records.LastAdded

   ' add record data
   record.Data("Name") = "Atlanta"
   record.Data("Pop") = 1200000
```

```
End Sub
```

CopyObjects (copy a selection to a new drawing)

```
'Copies objects selected in active drawing into a new drawing.
'Fields are NOT transferred.

Sub Main
   Dim windows, window, drawing, drawingTarget, object, objects, objectsTarget
   Set windows = Application.WindowSet

   ' ensure there is at least one opened window
   If windows.Count < 1 Then
      Exit Sub
   End If
   Set window = windows.ActiveWindow

   ' ensure active window contains drawing
   If window.ActiveComponent.Type <> ComponentDrawing Then
      Exit Sub
   End If
   Set drawing = window.ActiveComponent
   Set objects = drawing.Selection

   ' ensure there is at least one selected object
   If objects.Count < 1 Then
      Exit Sub
   End If

   ' create new drawing
   Set drawingTarget = document.NewDrawing(drawing.Name & " Selection")
   Set objectsTarget = drawingTarget.ObjectSet

   ' inherit coordinate system from original drawing
   drawingTarget.CoordinateSystem = drawing.CoordinateSystem

   ' batch updates for performance
   document.BatchUpdates = True

   ' copy all selected objects to newly created drawing
   For index = 0 To objects.Count-1
      Set object = objects(index)

      ' copy object
      objectsTarget.Add object.Geom
   Next

   ' flush accumulated updates
   document.BatchUpdates = False

   ' open created drawing
   drawingTarget.Open
End Sub
```

Forms (demonstrates the use of several basic Form controls)

```
' ************************************************************************

Sub Main
   Form.Visible = True
End Sub

' ************************************************************************

' -- add some items to list on startup
Sub Form_OnLoad
   List.AddItem "Item 1"
   List.AddItem "Item 2"
   List.AddItem "Item 3"
End Sub

' ************************************************************************

' -- add new item to list
Sub Add_Click
   List.AddItem Text.Text
End Sub

' ************************************************************************

' -- remove selected item from list if any
Sub Delete_Click
   If List.SelCount > 0 Then
      For i = List.ListCount-1 To 0 Step -1
         If  List.Selected(i) Then
            List.RemoveItem i
         End If
      Next
   End If
End Sub

' ************************************************************************

' -- remove all items from list
Sub DeleteAll_Click
   List.Clear
End Sub

' ************************************************************************

' -- close form
Sub Close_Click
   Form.Visible = False
End Sub

' ************************************************************************
```

Hello (Jscript, Pearl, Python, VBscript)

See http://www.activestate.com for more information

J-Script

```
// say hello using built-in Application object

function Main() {
   Application.MessageBox("Hello, World!");
}
```

Pearl Script

```
# say hello using built-in Application object
# Perl scripts require ActiveState ActivePerl version 1.0 or later.

sub Main {
   $Application->MessageBox("Hello, World!");
}
```

Python Script

```
#Python scripts require ActiveState ActivePython version 1.0 or later.

# say hello using built-in Application object

def Main():
   Application.MessageBox("Hello, World!")
```

VBscript

```
' -- say hello using native VBScript function

Sub Main
   Application.MessageBox "Hello, World!"
End Sub
```

Import Folder – VBscript (imports all MID/MIF pairs in a folder and sub-folders)

```
' ************************************************************************

// Ask user for starting folder and import it
function Main() {

   caption = "Batch Import";

   // query name of folder to import
   folder = Application.InputBox("Enter folder:", caption);
   if (!folder.length)
      return;

   // create file system object and obtain folder
   fso = new ActiveXObject("Scripting.FileSystemObject");
   fld = fso.GetFolder(folder);
   if (fld == null) {
      Application.MessageBox("Folder does not exist.", caption);
      return;
   }

   Import(fld);
}

' ************************************************************************

// Import folder
function Import(folder) {

   var imp = Application.NewImport("MIF");
   imp.ImportFormatting = true;

   // import files with "mif" extension
   var files = new Enumerator(folder.Files);
   for (; !files.atEnd(); files.moveNext()) {
      file = files.item();
      name = file.Name;
      dotPos = name.lastIndexOf(".");
      if (dotPos >= 0) {
         ext = name.substring(dotPos+1, name.length).toLowerCase();
         if (ext == "mif")
            imp.Import(file, PromptNone);
      }
   }

   // import subfolders
   var folders = new Enumerator(folder.SubFolders);
   for (; !folders.atEnd(); folders.moveNext())
      Import(folders.item());
}

' ************************************************************************
```

NDVI (computes Normalized Difference Vegetation Index – example of using the application Status Bar)

Computes NDVI (Normalized Difference Vegetation Index) using Band 1 and Band 2 and puts result into NDVI. Displays computation progress in status bar. Missing values are handled properly. Band 1 and and 2 are parts of the AVHRR import. Display Options for all three surfaces are set to render height with shading and autocontrast options turned off.

```
' Computes NDVI (Normalized Difference Vegetation Index) using Band 1
' and Band 2 and puts result into NDVI.

Sub Main

   ' obtain document and component set
   Dim document, components
   Set document = Application.ActiveDocument
   Set components = document.ComponentSet

   ' obtain pixel sets from both bands and resulting surface
   Dim band1, band1p, band2, band2p, ndvi, ndvip
   Set band1 = components(components.ItemByName("Band 1")).PixelSet
   Set band2 = components(components.ItemByName("Band 2")).PixelSet
   Set ndvi = components(components.ItemByName("NDVI")).PixelSet

   ' collapse updates until the entire operation is complete
   document.BatchUpdates = True

   ' compute NDVI displaying progress messages in status bar
   For i = 0 To ndvi.Count-1
      Set band1p = band1(i)
      Set band2p = band2(i)
      Set ndvip = ndvi(i)

      If band1p.IsMissing Then
         ndvip.Mask = 128      ' make pixel invisible
      ElseIf band2p.IsMissing Then
         ndvip.Mask = 128      ' make pixel invisible
      Else
         ndvip.Mask = 0         ' make pixel visible
         ndvip.Value = CDbl(band2p.Value - band1p.Value) / CDbl(band2p.Value +
band1p.Value)
      End If

      If i Mod 100 = 0 Then
         Application.StatusText = CStr(i) + " out of " + CStr(ndvi.Count) + " pixels
processed."
      End If
   Next

   ' flush update queue
```

```
   document.BatchUpdates = False

   ' open computed NDVI surface
   components(components.ItemByName("NDVI")).Open
End Sub
```

Quartiles (assigns values in the given numeric column to quartiles)

```
' Assigns values in the given numeric column to quartiles, expressed
' as a value from 1 to 4 in a new integer column.

Sub Main
   ' ensure there is active window
   Set windows = Application.WindowSet
   If windows.Count = 0 Then
      Exit Sub
   End If
   Set window = windows.ActiveWindow
   Set component = window.Component

   ' ensure active component is table
   If component.Type <> ComponentTable Then
      Exit Sub
   End If

   ' ensure table contains at least one record
   Set records = component.RecordSet
   If records.Count = 0 Then
      Exit Sub
   End If

   ' query for field name
   columnName = InputBox("Enter numeric column name:", "Quartiles",
window.ActiveColumn.Name)
   If Trim(CStr(columnName)) = "" Then
      Exit Sub
   End If

   ' check of column type and name
   columnIndex = component.columnSet.ItemByName(columnName)
   If columnIndex < 0 Then
      Application.MessageBox "Can't find column '" & columnName & "'."
      Exit Sub
   End If
   Set column = component.ColumnSet(columnIndex)
   If Not column.IsTypeNumeric Then
      Application.MessageBox "Column '" & columnName & "' is not numeric."
      Exit Sub
   End If

   ' create new column or reuse existing column
   quartileName = columnName + " Quartile"
   columnIndex = component.columnSet.ItemByName(quartileName)
   createNew = False
```

```
   If columnIndex < 0 Then
      createNew = True
   ElseIf Not component.columnSet(columnIndex).IsTypeNumeric Then
      createNew = True
   End If
   If createNew Then
      Set columnNew = component.ColumnSet.NewColumn
      columnNew.Name = quartileName
      columnNew.Type = ColumnTypeInt32
      component.ColumnSet.Add(columnNew)
      quartileName = component.ColumnSet.LastAdded.Name
   End If

   ' compute frequency breaks
   minValue = records.Minimum(columnName).Item(0).Data(columnName)
   maxValue = records.Maximum(columnName).Item(0).Data(columnName)
   range = (maxValue - minValue)/4
   R1 = minValue + range
   R2 = R1 + range
   R3 = R2 + range

   ' process records
   For index = 0 To records.Count-1
      Set record = records(index)

      value = record.Data(columnName)
      If value > R3 Then
         record.Data(quartileName) = 4
      ElseIf value > R2 Then
         record.Data(quartileName) = 3
      ElseIf value > R1 Then
         record.Data(quartileName) = 2
      Else
         record.Data(quartileName) = 1
      End If
   Next
End Sub
```

Random Points (creates a drawing with points randomly taken from "Data Table")

```
'Create Random Points

' Creates a drawing with points taken from the "Data" table.
' Point locations are random.

Sub Main
   Set components = document.ComponentSet

   ' -- protect from being run for the second time
   If components.ItemByName("Data Drawing") >= 0 Then
      Application.MessageBox "'Data Drawing' component already exists.  Delete it and
re-run the script."
      Exit Sub
   End If
```

```
   ' -- create target drawing
   Set drawing = document.NewDrawing("Data Drawing",
Application.DefaultCoordinateSystemLatLon)

   Set objects = drawing.ObjectSet
   Set table = drawing.OwnedTable
   Set columns = table.ColumnSet
   Set records = table.RecordSet

   ' -- obtain source data readers
   Set data = components(components.ItemByName("Data"))
   Set dataColumns = data.ColumnSet
   Set dataRecords = data.RecordSet

   ' -- walk through data columns creating similar columns in target table
   For columnIndex = 0 To dataColumns.Count-1
      Set dataColumn = dataColumns(columnIndex)

      Set column = columns.NewColumn()
      column.Name = dataColumn.Name
      column.Type = dataColumn.Type
      column.Size = dataColumn.Size
      columns.Add(column)
   Next

   ' -- walk through data records creating random points in target drawing
   For recordIndex = 0 To dataRecords.Count-1

      ' -- create random lat/lon location
      Set point = Application.NewPoint
      point.X = (Rnd-0.5)*360
      point.Y = (Rnd-0.5)*180

      ' -- create drawing object
      objects.Add Application.NewGeom(GeomPoint, point)

      Set dataRecord = dataRecords(recordIndex)

      ' -- transfer fields
      Set record = records(records.Count-1)
      For columnIndex = 0 To dataColumns.Count-1
         Set dataColumn = dataColumns(columnIndex)

         record.Data(dataColumn.Name) = dataRecord.Data(dataColumn.Name)
      Next
   Next

   drawing.Open
End Sub
```

Sequential Lines (joins points selected within the active drawing window with a line)

```
' Join Selected Points

' Joins points selected within the active drawing window with a
' line.  The order of points on a line is determined by their IDs.

Sub Main
   Set windows = Application.WindowSet

   ' -- fail if there is no opened windows
   If windows.Count = 0 Then
      Application.MessageBox "No active window"
      Exit Sub
   End If

   Set window = windows.ActiveWindow
   Set component = window.Component

   ' -- fail if active component is not a drawing
   If component.Type <> ComponentDrawing Then
      Application.MessageBox "Active component is not a drawing"
      Exit Sub
   End If

   ' -- create temporary query component
   Set components = document.ComponentSet
   Set query = document.NewQuery("Temp")
   query.Text = "SELECT ID, [X (I)], [Y (I)] FROM [" + component.Name + "] WHERE
[Selection (I)] AND [Type (I)] = 1 ORDER BY ID;"
   ' -- KLUDGE: verbatim value of 1 in '[Type (I)] = 1' fragment filters
   ' --   out line and area objects
   Set table = query.Table

   ' -- ensure there are at least two records
   Set records = table.RecordSet
   If  records.Count < 2 Then
      ' -- remove temporary query component
      components.Remove(components.ItemByID(query.ID))

      Application.MessageBox "There should be at least two selected points"
      Exit Sub
   End If

   ' -- create new point set object
   Set pointSet = Application.NewPointSet

   ' -- scan queried table adding locations to point set
   For recordIndex = 0 to records.Count-1
      Set record = records(recordIndex)
      Set point = Application.NewPoint
      point.X = CDbl(record.Data("X (I)"))
      point.Y = CDbl(record.Data("Y (I)"))
      pointSet.Add point
   Next
```

```
   ' -- create new geometric entity
   Set geom = Application.NewGeom(GeomLine, pointSet)

   ' -- create new line object
   component.ObjectSet.Add geom

   ' -- remove temporary query component
   components.Remove(components.ItemByID(query.ID))
End Sub
```

Spatial Operations (create a circle around each city)

Create a Circle Around Each City

```
' Creates a circle around each city taking radius of the circle from table.

Sub Main
   Set components = document.ComponentSet

   ' locate city drawing
   drawingIndex = components.ItemByName("Cities")
   If drawingIndex < 0 Then
      Application.MessageBox "No 'Cities' component."
      Exit Sub
   End If
   Set drawing = components(drawingIndex)

   ' ensure component type is drawing
   If drawing.Type <> ComponentDrawing Then
      Application.MessageBox "'Cities' is not a drawing."
      Exit Sub
   End If

   ' obtain set of drawing objects
   Set objects = drawing.ObjectSet

   ' ensure drawing is not empty
   If objects.Count <= 0 Then
      Application.MessageBox "No objects in 'Cities'."
      Exit Sub
   End If

   ' obtain owned table and set of table records
   Set table = drawing.OwnedTable
   Set records = table.RecordSet

   ' ensure table contains radius column
   radiusIndex = table.ColumnSet.ItemByName("Radius (km)")
   If radiusIndex < 0 Then
      Application.MessageBox "No 'Radius (km)' column."
      Exit Sub
   End If
```

```
   Pi = 3.141592653589793

   Dim pointIDs
   ReDim pointIDs(objects.Count-1)
   pointCount = 0

   ' cache IDs of point objects
   For objectIndex = 0 To objects.Count-1
      Set object = objects(objectIndex)
      If object.Type = ObjectPoint Then
         pointIDs(pointCount) = object.ID
         pointCount = pointCount + 1
      End If
   Next
   If pointCount = 0 Then
      Application.MessageBox "No points in 'Cities'."
      Exit Sub
   End If

   ' <<< batch updates here if the number of cities is too large

   ' create circle for each point object
   For pointIndex = 0 To pointCount-1
      Set object = objects(objects.ItemByID(pointIDs(pointIndex)))
      Set record = records(records.ItemByID(pointIDs(pointIndex)))

      ' obtain city location
      Dim location
      Set location = object.Geom.Center

      ' obtain radius of circle from table
      radius = 1000 * CDbl(record.Data("Radius (km)"))

      ' create circle
      Set pointSet = Application.NewPointSet

      For i = 0 To 31
         Set point = Application.NewPoint

         ' calculate angle
         angle = i*2*Pi/32

         ' obtain point of circle
         point.X = location.X + radius*Cos(angle)
         point.Y = location.Y + radius*Sin(angle)
         pointSet.Add(point)
      Next

      objects.Add Application.NewGeom(GeomArea, pointSet)
   Next
End Sub
```

Transefer County Name to Each City

```
' Equips each city with name of the country it is in.

Sub Main
   Set components = document.ComponentSet

   ' locate city drawing
   drawingIndex = components.ItemByName("Cities")
   If drawingIndex < 0 Then
      Application.MessageBox "No 'Cities' component."
      Exit Sub
   End If
   Set cityDrawing = components(drawingIndex)

   ' locate country drawing
   drawingIndex = components.ItemByName("Countries")
   If drawingIndex < 0 Then
      Application.MessageBox "No 'Countries' component."
      Exit Sub
   End If
   Set countryDrawing = components(drawingIndex)

   ' ensure both components are drawings
   If cityDrawing.Type <> ComponentDrawing Then
      Application.MessageBox "'Cities' is not a drawing."
      Exit Sub
   End If
   If countryDrawing.Type <> ComponentDrawing Then
      Application.MessageBox "'Countries' is not a drawing."
      Exit Sub
   End If

   ' obtain set of cities and set of countries
   Set cities = cityDrawing.ObjectSet
   Set countries = countryDrawing.ObjectSet

   ' ensure both object sets are not empty
   If cities.Count <= 0 Then
      Application.MessageBox "No objects in 'Cities'."
      Exit Sub
   End If
   If countries.Count <= 0 Then
      Application.MessageBox "No objects in 'Countries'."
      Exit Sub
   End If

   ' obtain owned tables and set of table records
   Dim cityTable, cityRecords, cityRecord, countryTable, countryRecords, countryRecord
   Set cityTable = cityDrawing.OwnedTable
   Set cityRecords = cityTable.RecordSet
   Set countryTable = countryDrawing.OwnedTable
   Set countryRecords = countryTable.RecordSet

   ' ensure both tables contain country column
   If cityTable.ColumnSet.ItemByName("Country") < 0 Then
```

```
      Application.MessageBox "No 'Country' column in 'Cities'."
      Exit Sub
   End If
   If countryTable.ColumnSet.ItemByName("Country") < 0 Then
      Application.MessageBox "No 'Country' column in 'Countries'."
      Exit Sub
   End If

   ' <<< batch updates here if the number of cities is large

   Application.StatusText = "Cleaning up cities."

   ' cleanup cities
   For cityIndex = 0 To cities.Count-1
      Set city = cities(cityIndex)
      Set cityRecord = cityRecords(cityRecords.ItemByID(city.ID))

      ' clean up country field
      cityRecord.Data("Country") = ""

      ' <<< cleanup other fields here if necessary
   Next

   Application.StatusText = "Transferring data."

   ' transfer columns from countries to cities going by countries so that each country
gets
   ' simplified only once
   For countryIndex = 0 To countries.Count-1
      Application.StatusText = CStr(countryIndex+1) + " of " + CStr(countries.Count) +
" countries."

      Set country = countries(countryIndex)
      Set countryRecord = countryRecords(countryRecords.ItemByID(country.ID))

      ' skip point and line objects that can get created by the user
      If country.Type = ObjectArea Then

         ' transfer columns from current country to all cities that lie within this
country
         For cityIndex = 0 To cities.Count-1
            Set city = cities(cityIndex)

            ' skip area objects that can get created by the CreateCircle... script or
by the user
            If city.Type = ObjectPoint Then
               Set cityRecord = cityRecords(cityRecords.ItemByID(city.ID))

               ' protect from double-visiting the same city
               If cityRecord.Data("Country") = "" Then

                  ' <<< project city here IF coordinate systems of country drawing and
city drawing are different

                  If country.Geom.CheckContains(city.Geom) Then

                     ' copy country field
```

```
                    cityRecord.Data("Country") = countryRecord.Data("Country")

                    ' <<< transfer other fields here if necessary

                    ' stop crawling through countries
                End If
            End If
        End If
      Next
   End If
Next

Application.StatusText = ""
End Sub
```

Transfer Country Name to Each City using Query

```
' Equips each city with name of the country it is in using queries.

Sub Main
   Set components = document.ComponentSet

   ' locate city drawing
   drawingIndex = components.ItemByName("Cities")
   If drawingIndex < 0 Then
      Application.MessageBox "No 'Cities' component."
      Exit Sub
   End If
   Set drawing = components(drawingIndex)

   ' ensure component is drawing
   If drawing.Type <> ComponentDrawing Then
      Application.MessageBox "'Cities' is not a drawing."
      Exit Sub
   End If

   ' obtain table and set of table records
   Set table = drawing.OwnedTable
   Set records = table.RecordSet

   Application.StatusText = "Cleaning up cities."

   ' cleanup cities
   For recordIndex = 0 To records.Count-1
      Set record = records(recordIndex)

      ' clean up country field
      record.Data("Country") = ""
   Next

   ' locate service query
   queryIndex = components.ItemByName("Service Query")
   If queryIndex < 0 Then
      Application.MessageBox "No 'Service Query' component."
```

```
      Exit Sub
   End If
   Set query = components(queryIndex)

   ' ensure component is drawing
   If query.Type <> ComponentQuery Then
      Application.MessageBox "'Service Query' is not a query."
      Exit Sub
   End If

   Application.StatusText = "Executing query."

   ' obtain query table and set of table records
   Set table = query.Table
   Set records = table.RecordSet

   Application.StatusText = "Transferring data."

   ' transfer data from one column to another
   For recordIndex = 0 To records.Count-1
      Set record = records(recordIndex)

      ' copy country field
      record.Data("Country") = record.Data("CountryOrg")
   Next

   Application.StatusText = ""
End Sub
```

Surfaces (computes average height and selects pixels below this height)

Select Band

```
Sub Main
   Set app = Application

   ' -- locate opened window and ensure it displays surface
   Set window = app.WindowSet.ActiveWindow
   Set component = window.Component
   If  window.Component.Type <> ComponentSurface Then
      Application.MessageBox "No surface."
      Exit Sub
   End If

   Set surface = window.Component
   Set pixels = surface.PixelSet

   ' -- input a pair of values
   height1 = app.InputBox("Value 1:", "Select Band")
   If height1 = "" Then
      Exit Sub
   End If
   height2 = app.InputBox("Value 2:", "Select Band")
```

```
   If height2 = "" Then
      Exit Sub
   End If
   If CDbl(height1) < CDbl(height2) Then
      heightLow = CDbl(height1)
      heightHgh = CDbl(height2)
   Else
      heightLow = CDbl(height2)
      heightHgh = CDbl(height1)
   End If

   app.StatusText = "Selecting pixels"

   ' -- turn batch updates on
   app.ActiveDocument.BatchUpdates = True

   ' -- select pixels
   For i = 0 To pixels.Count-1
      If i Mod 100 = 99 Then
         app.StatusText = "Selecting pixels (" & CStr(i+1) & " of " &
CStr(pixels.Count+1) & ")"
      End If
      Set pixel = pixels(i)
      If Not pixel.IsMissing Then
         If pixel.Value >= heightLow And pixel.Value <= heightHgh Then
            pixel.Mask = pixel.Mask Or 1
         Else
            pixel.Mask = pixel.Mask And Not 1
         End If
      End If
   Next

   ' -- turn batch updates off
   app.ActiveDocument.BatchUpdates = False

   app.StatusText = "Done"
End Sub
```

Select Lower Part

```
Sub Main
   Set app = Application

   ' -- locate opened window and ensure it displays surface
   Set window = app.WindowSet.ActiveWindow
   Set component = window.Component
   If  window.Component.Type <> ComponentSurface Then
      Application.MessageBox "No surface."
      Exit Sub
   End If

   Set surface = window.Component
   Set pixels = surface.PixelSet
```

```
   app.StatusText = "Computing average height"

   ' -- compute average height
   height = 0
   heightPixels = 0
   For i = 0 To pixels.Count-1
      If i Mod 100 = 99 Then
         app.StatusText = "Computing average height (" & CStr(i+1) & " of " &
CStr(pixels.Count+1) & ")"
      End If
      Set pixel = pixels(i)
      If Not pixel.IsMissing Then
         height = height + CDbl(pixel.Value)
         heightPixels = heightPixels + 1
      End If
   Next
   If heightPixels = 0 Then
      Application.MessageBox "All surface pixels are invisible."
      Exit Sub
   End If
   height = CDbl(height) / CDbl(heightPixels)

   app.StatusText = "Selecting pixels"

   ' -- turn batch updates on
   app.ActiveDocument.BatchUpdates = True

   ' -- select pixels
   For i = 0 To pixels.Count-1
      If i Mod 100 = 99 Then
         app.StatusText = "Selecting pixels (" & CStr(i+1) & " of " &
CStr(pixels.Count+1) & ")"
      End If
      Set pixel = pixels(i)
      If Not pixel.IsMissing Then
         If pixel.Value < height Then
            pixel.Mask = pixel.Mask Or 1
         Else
            pixel.Mask = pixel.Mask And Not 1
         End If
      End If
   Next

   ' -- turn batch updates off
   app.ActiveDocument.BatchUpdates = False

   app.StatusText = "Done"
End Sub
```

Select Slope

```
Sub Main
   Set app = Application
```

```
   ' -- locate opened window and ensure it displays surface
   Set window = app.WindowSet.ActiveWindow
   Set component = window.Component
   If  window.Component.Type <> ComponentSurface Then
      Application.MessageBox "No surface."
      Exit Sub
   End If

   Set surface = window.Component
   Set pixels = surface.PixelSet
   Set coordSystem = surface.CoordinateSystem
   Set coordParameters = coordSystem.Parameters
   scaleX = coordParameters(coordParameters.ItemByName("localScaleX")).Value
   scaleY = coordParameters(coordParameters.ItemByName("localScaleY")).Value
   slopeMax = 25
   width = surface.Width

   app.StatusText = "Selecting pixels"

   ' -- turn batch updates on
   app.ActiveDocument.BatchUpdates = True

   ' -- select pixels
   For i = 0 To pixels.Count-1
      If i Mod 100 = 99 Then
         app.StatusText = "Selecting pixels (" & CStr(i+1) & " of " &
CStr(pixels.Count+1) & ")"
      End If

      ' the following code assumes that Z value is measured in the same unit as X and
Y

      pixelX = i Mod width
      pixelY = i \ width

      Set pixel = pixels(i)
      If  pixel.IsMissing() Then
         pixel.Mask = pixel.Mask And Not 1
      Else
         zE = pixel.Value

         ' -- compute height in eight neighboring pixels
         zA = PixelZ(surface, pixels, pixelX-1, pixelY-1, zE)
         zB = PixelZ(surface, pixels, pixelX, pixelY-1, zE)
         zC = PixelZ(surface, pixels, pixelX+1, pixelY-1, zE)
         zD = PixelZ(surface, pixels, pixelX-1, pixelY, zE)
         zF = PixelZ(surface, pixels, pixelX+1, pixelY, zE)
         zG = PixelZ(surface, pixels, pixelX-1, pixelY+1, zE)
         zH = PixelZ(surface, pixels, pixelX, pixelY+1, zE)
         zI = PixelZ(surface, pixels, pixelX+1, pixelY+1, zE)

         ' -- compute deltas in X and Y directions
         deltaX1 = zC - zA
         deltaX2 = zF - zD
         deltaX3 = zI - zG
         deltaY1 = zA - zG
         deltaY2 = zB - zH
```

```
         deltaY3 = zC - zI
         deltaX = (deltaX1 + deltaX2 + deltaX3) / 3 * scaleX
         deltaY = (deltaY1 + deltaY2 + deltaY3) / 3 * scaleY

         ' -- compute slope
         slope = Sqr(deltaX^2 + deltaY^2)/2

         ' -- translate slope to percents
         If slope > 1 Then
            slope = 200 - 100 / slope
         Else
            slope = slope * 100
         End If
         If slope < slopeMax Then
            pixel.Mask = pixel.Mask Or 1
         Else
            pixel.Mask = pixel.Mask And Not 1
         End If
      End If
   Next

   ' -- turn batch updates off
   app.ActiveDocument.BatchUpdates = False

   app.StatusText = "Done"
End Sub

' ****************************************************************************

' -- compute height of given pixel
Function PixelZ(surface, pixels, pixelX, pixelY, z)
   If pixelX < 0 Then
      PixelZ = z
   ElseIf pixelY < 0 Then
      PixelZ = z
   ElseIf pixelX >= surface.Width Then
      PixelZ = z
   ElseIf pixelY >= surface.Height Then
      PixelZ = z
   Else
      Set pixel = pixels(pixelX + pixelY*surface.Width)
      If  pixel.IsMissing() Then
         PixelZ = z
      Else
         PixelZ = pixel.Value
      End If
   End If
End Function
```

Table Generic (selects all records with value in an active column equal to that in the active cell)

Select Same Year-Month to Active

```
Sub Main
   Set windows = Application.WindowSet
   Set window = windows.ActiveWindow
   Set component = window.Component
   If component.Type = ComponentTable Then

      ' check if active column is Date
      Set column = window.ActiveColumn
      If  column.Type = ColumnTypeTime Then

         Set record = window.ActiveRecord
         valueActive = CDate(record.Data( column.ID ))
         monthActive = DatePart("m", valueActive)
         yearActive = DatePart("yyyy", valueActive)

         ' loop through all records selecting those with same month
         Set records = component.RecordSet
         For nItem = 0 To records.Count-1
            Set record = records( nItem )
            value = CDate(record.Data( column.ID ))

            If DatePart("m", value) = monthActive And DatePart("yyyy", value) =
yearActive Then
               record.Mask = record.Mask Or 1
            Else
               record.Mask = record.Mask And Not 1
            End If
         Next
      End If
   End If
End Sub
```

Select Same to Active

```
Sub Main
   Set windows = Application.WindowSet
   Set window = windows.ActiveWindow
   Set component = window.Component
   If component.Type = ComponentTable Then

      ' reset selection
      Set records = component.RecordSet
      For nItem = 0 To records.Count-1
         Set record = records( nItem )
         record.Mask = record.Mask And Not 1
      Next
```

```
      ' create selection
      Set column = window.ActiveColumn
      Set record = window.ActiveRecord
      value = record.Data( column.ID )
      Set selections = records.EqualTo( column.ID, value )
      For nItem = 0 to selections.Count-1
         Set record = selections( nItem )
         record.Mask = record.Mask Or 1
      Next
   End If
End Sub
```

Table Nwind (adds a relation between Categories and Products)

Add Relation between Categories and Products

```
Sub Main
   ' initialization
   set components = document.ComponentSet

   ' get categories key column
   nItem = components.ItemByName( "Categories" )
   set tableCategories = components.Item( nItem )
   set columnsCategories = tableCategories.ColumnSet
   nItem = columnsCategories.ItemByName( "Category ID" )
   set columnCategoriesKey = columnsCategories.Item( nItem )

   ' get products key column
   nItem = components.ItemByName( "Products" )
   set tableProducts = components.Item( nItem )
   set columnsProducts = tableProducts.ColumnSet
   nItem = columnsProducts.ItemByName( "Category ID" )
   set columnProductsKey = columnsProducts.Item( nItem )

   ' add relation
   set relations = tableCategories.RelationSet
   relations.Add columnCategoriesKey, columnProductsKey

   ' map products column
   set columnProductsMap = columnsCategories.NewColumn
   nItem = columnsProducts.ItemByName( "Product Name" )
   columnProductsMap.Name = "Sample Product"
   columnProductsMap.OriginColumn = columnsProducts.Item( nItem )
   columnsCategories.Add(columnProductsMap)

   ' map categories column
   set columnCategoriesMap = columnsProducts.NewColumn
   nItem = columnsCategories.ItemByName( "Category Name" )
   columnCategoriesMap.Name = "Category Name"
   columnCategoriesMap.OriginColumn = columnsCategories.Item( nItem )
   columnsProducts.Add(columnCategoriesMap)

   tableCategories.Open
```

```
   tableProducts.Open
End Sub
```

Clean Up Opened Table

```
Sub Main
   Set windows = Application.WindowSet
   Set window = windows.ActiveWindow
   Set component = window.Component
   If component.Type = ComponentTable Then

      ' remove relations
      Set relations = component.RelationSet
      For nItem = relations.Count-1 To 0 Step -1
         relations.Remove nItem
      Next

      ' remove atoms
      Set atoms = component.DSSAtomSet
      For nItem = atoms.Count-1 To 0 Step -1
         atoms.Remove nItem
      Next

      ' reset selection
      Set records = component.RecordSet
      For nItem = 0 To records.Count-1
         Set record = records( nItem )
         record.Mask = record.Mask And Not 1
      Next

   End If
End Sub
```

Sales by Cities Report

```
Sub Main
   ' create temporary query
   Dim s
   s = "select [Customers].[City], " & _
      "sum([Order Details].[Unit Price]*[Order Details].[Quantity]*(100-[Order
Details].[Discount])/100) as [Total] " & _
      "from [Customers], [Orders], [Order Details] " & _
      "where [Customers].[Customer ID]=[Orders].[Customer ID] " & _
      "and [Orders].[Order ID] = [Order Details].[Order ID] " & _
      "group by [Customers].[City]"

   set query = document.NewQuery( "Temporary Query - Sales by Cities" )
   query.Text = s
   set table = query.Table
   set records = table.RecordSet
```

```
   ' create new table
   set columns = Application.NewColumnSet
   set column = columns.NewColumn
   column.Name = "City ID"
   column.Type = ColumnTypeInt32U
   columns.Add column
   column.Name = "City Name"
   column.Type = ColumnTypeWText
   columns.Add column
   column.Name = "Total"
   column.Type = ColumnTypeCurrency
   column.Format.Align = ColumnAlignRight
   column.Format.Decimals = 2
   column.Format.Style = "$ (dollar)"
   column.Format.Width = 100
   columns.Add column
   set tableCities = document.NewTable( "Sales by Cities Report Table", columns )
   set recordsCities = tableCities.RecordSet
   set recordNew = recordsCities.NewRecord

   ' create new comments
   set commentCities = document.NewComments( "Sales by Cities Report Text" )
   commentCities.AddText "Sales by Cities" & vbCRLF & vbCRLF

   ' fill out new table and new comments
   for nItem = 0 to records.Count-1
      set record = records( nItem )
      recordNew.Data( "City ID" ) = nItem + 1
      recordNew.Data( "City Name") = record.Data( "City" )
      recordNew.Data( "Total") = record.Data( "Total" )
      recordsCities.Add recordNew

      v = record.Data( "City" )
      commentCities.AddText v
      commentCities.AddText ": "
      v = record.Data( "Total" )
      commentCities.AddText FormatCurrency(CCur(v), 2, True)
      commentCities.AddText vbCRLF
   next
   tableCities.Open
   commentCities.Open

   ' remove temporary query
   set components = document.ComponentSet
   nItem = components.ItemByName( "Temporary Query - Sales by Cities" )
   components.Remove(nItem)
End Sub
```

Select Products by User Category

```
Sub Main
   ' initialization
   set components = document.ComponentSet
```

```
   ' get category id
   nItem = components.ItemByName( "Categories" )
   set tableCategories = components.Item( nItem )
   set recordsCategories = tableCategories.RecordSet
   category = Application.InputBox( "Enter category name:", "Select Products of User
Category", "Seafood" )
   If category = "" Then
      Exit Sub
   End If
   nItem = recordsCategories.ItemByValue( "Category Name", category )
   set record = recordsCategories.Item( nItem )
   value = record.Data( "Category ID" )

   ' reset selection in Products table
   nItem = components.ItemByName( "Products" )
   set tableProducts = components.Item( nItem )
   set recordsProducts = tableProducts.RecordSet
   For nItem = 0 To recordsProducts.Count-1
      Set record = recordsProducts( nItem )
      record.Mask = record.Mask And Not 1
   Next

   ' select products with selected category
   Set selections = recordsProducts.EqualTo( "Category ID", value )
   For nItem = 0 To selections.Count-1
      Set record = selections( nItem )
      record.Mask = record.Mask Or 1
   Next

   tableProducts.Open
End Sub
```

Select Young Yet Experienced Employee

```
Sub Main
   ' initialization
   set components = document.ComponentSet
   nItem = components.ItemByName( "Employees" )
   set table = components.Item( nItem )
   set columns = table.ColumnSet
   set column = columns.NewColumn
   set records = table.RecordSet
   set atoms = table.DSSAtomSet
   set entries = column.DSSQuery

   ' create Young atom
   nItem = columns.ItemByName( "Birth Date" )
   set columnBirthDate = columns.Item( nItem )
   set atomYoung = atoms.NewDSSAtom
   atomYoung.Name = "Young"
   atomYoung.Column = columnBirthDate
   atomYoung.MakeHigh 0, -1
   atoms.Add( atomYoung )
   set atomYoung = atoms.LastAdded
```

```
   ' create Young link
   set entryYoung = entries.NewDSSQueryEntry
   entryYoung.Atom = atomYoung
   entryYoung.Junction = DSSJunctionAnd
   entries.Add entryYoung

   ' create Experienced atom
   nItem = columns.ItemByName( "Hire Date" )
   set columnHireDate = columns.Item( nItem )
   set atomExperienced = atoms.NewDSSAtom
   atomExperienced.Name = "Experienced"
   atomExperienced.Column = columnHireDate
   atomExperienced.MakeLow 0, -1
   atoms.Add( atomExperienced )
   set atomExperienced = atoms.LastAdded

   ' create Experienced link
   set entryExperienced = entries.NewDSSQueryEntry
   entryExperienced.Atom = atomExperienced
   entryExperienced.Junction = DSSJunctionAnd
   entries.Add entryExperienced

   ' create temporary column
   column.Name = "Golden Mean"
   columns.Add( column )
   set column = columns.LastAdded

   ' reset selection
   for nItem = 0 to records.Count-1
      set record = records( nItem )
      record.Mask = record.Mask and not 1
   next

   ' create selection
   set selections = records.Maximum( column.ID )
   for nItem = 0 to selections.Count-1
      set record = selections( nItem )
      record.Mask = record.Mask or 1
   next

   ' remove atoms and temporary column
   nItem = atoms.ItemByID( atomYoung.ID )
   atoms.Remove( nItem )
   nItem = atoms.ItemByID( atomExperienced.ID )
   atoms.Remove( nItem )

   table.Open
End Sub
```

Table Nwind Forms (walks through categories displaying list of products for current category)

Category Viewer




```
' ************************************************************************

' -- launch form on startup
Sub Main
   Form.Visible = True
End Sub

' ************************************************************************

Dim Categories
CategoryIndex = -1
CategoryCount = 10

' --------------------------

' -- jump to first category on load

' ************************************************************************

Sub Form_OnLoad

   ' -- set up global variables
   Set Components = Application.ActiveDocument.ComponentSet
   CatPos = Components.ItemByName("Categories")
   If CatPos < 0 Then
      Application.MessageBox "'Categories' table not found."
      Form.Visible = False
   End If
   Set Categories = Components(CatPos)
   CategoryIndex = 0
   CategoryCount = Categories.RecordSet.Count

   ' -- load controls
   Reload
End Sub
```

```
' ************************************************************************

' -- close form
Sub Close_Click
   Form.Visible = False
End Sub

' ************************************************************************

' -- jump to first category
Sub First_Click
   If CategoryIndex <> 0 Then
      CategoryIndex = 0
      Reload
   End If
End Sub

' ************************************************************************

' -- jump to last category
Sub Last_Click
   If CategoryIndex <> CategoryCount-1 Then
      CategoryIndex = CategoryCount-1
      Reload
   End If
End Sub

' ************************************************************************

' -- jump to next category if any
Sub Next_Click
   If CategoryIndex < CategoryCount-1 Then
      CategoryIndex = CategoryIndex+1
      Reload
   End If
End Sub

' ************************************************************************

' -- jump to previous category if any
Sub Previous_Click
   If CategoryIndex > 0 Then
      CategoryIndex = CategoryIndex-1
      Reload
   End If
End Sub

' ************************************************************************

' -- load current category
Sub Reload
   Set Rec = Categories.RecordSet(CategoryIndex)

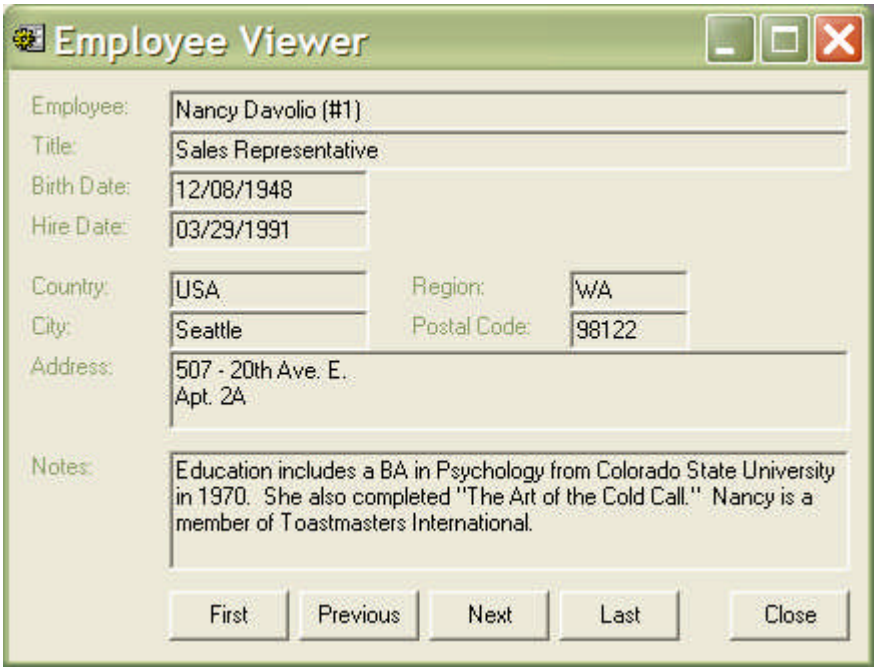
   ' -- set category readout
   Category.Text = Rec.Data("Category Name") & " (#" & CStr(CategoryIndex+1) & ")"
```

```
   Set Components = Application.ActiveDocument.ComponentSet
   Set TargetTable = Components(Components.ItemByName("Products"))
   Set TargetRecords = TargetTable.RecordSet.EqualTo("Category ID", Rec.Data("Category
ID"))

   ' -- fill product list
   Products.Clear
   For Target = 0 To TargetRecords.Count-1
      Products.AddItem CStr(TargetRecords(Target).Data("Product Name"))
   Next
End Sub

' ********************************************************************************
```

Employee Viewer




```
' ********************************************************************************

' -- launch form on startup
Sub Main
   Form.Visible = True
End Sub

' ---------------------------

Dim Employees
EmployeeIndex = -1
EmployeeCount = 10

' ---------------------------
```

```
' ************************************************************************

' -- jump to first employee on load
Sub Form_OnLoad

   ' -- set up global variables
   Set Components = Application.ActiveDocument.ComponentSet
   EmpPos = Components.ItemByName("Employees")
   If EmpPos < 0 Then
      Application.MessageBox "'Employees' table not found."
      Form.Visible = False
   End If
   Set Employees = Components(EmpPos)
   EmployeeIndex = 0
   EmployeeCount = Employees.RecordSet.Count

   ' -- load controls
   Reload
End Sub

' ************************************************************************

' -- close form
Sub Close_Click
   Form.Visible = False
End Sub

' ************************************************************************

' -- jump to first employee
Sub First_Click
   If EmployeeIndex <> 0 Then
      EmployeeIndex = 0
      Reload
   End If
End Sub

' ************************************************************************

' -- jump to last employee
Sub Last_Click
   If EmployeeIndex <> EmployeeCount-1 Then
      EmployeeIndex = EmployeeCount-1
      Reload
   End If
End Sub

' ************************************************************************

' -- jump to next employee if any
Sub Next_Click
   If EmployeeIndex < EmployeeCount-1 Then
      EmployeeIndex = EmployeeIndex+1
      Reload
   End If
End Sub
```

```
' ************************************************************************

' -- jump to previous employee if any
Sub Previous_Click
   If EmployeeIndex > 0 Then
      EmployeeIndex = EmployeeIndex-1
      Reload
   End If
End Sub

' ************************************************************************

' -- load current employee
Sub Reload
   Set Rec = Employees.RecordSet(EmployeeIndex)

   ' -- set employee readouts
   Employee.Text = Rec.Data("First Name") & " " & Rec.Data("Last Name") & " (#" &
CStr(EmployeeIndex+1) & ")"
   Title.Text = Rec.Data("Title")
   BirthDate.Text = Rec.Data("Birth Date")
   HireDate.Text = Rec.Data("Hire Date")
   Country.Text = Rec.Data("Country")
   Region.Text = Rec.Data("Region")
   City.Text = Rec.Data("City")
   PostalCode.Text = Rec.Data("Postal Code")
   Address.Text = Rec.Data("Address")
   Notes.Text = Rec.Data("Notes")
End Sub

' ************************************************************************
```

Text Formats (utilities for reading data from simple ASCII formats)

Export Released45 Text File

```
' Exports selected objects in opened drawing to a file that can be
' read by Release 4.5 "Read Text File" solver.

Sub Main
   Set Windows = Application.WindowSet

   ' fail if there is no opened windows
   If Windows.Count = 0 Then
      Application.MessageBox "No active window"
      Exit Sub
   End If

   Set Window = Windows.ActiveWindow
   Set Component = Window.Component
```

```
   ' fail if active component is not a drawing
   If Component.Type <> ComponentDrawing Then
      Application.MessageBox "Active component is not a drawing"
      Exit Sub
   End If

   Set ObjSelection = Component.Selection

   ' fail if there's no selected objects
   If ObjSelection.Count < 1 Then
      Application.MessageBox "No selected objects"
      Exit Sub
   End If

   ' ask for output filename
   FileName = InputBox("File name", "Input")
   Set FSO = CreateObject("Scripting.FileSystemObject")

   ' create file
   Set Stream = FSO.CreateTextFile(FileName, True)

   ' write all selected objects
   For i = 0 to ObjSelection.Count - 1
      Set ObjPointSet = ObjSelection(i).Geom.BranchSet(0).PointSet

      ' write object type / number of points
      Str = ""
      If ObjSelection.Item(i).Type = ObjectPoint Then
         Str = "1"
      ElseIf ObjSelection.Item(i).Type = ObjectLine Then
         Str = "2"
      Else
         Str = "3"
      End If
      Stream.WriteLine(Str & " " & ObjPointSet.Count)

      ' write first object branch
      For k = 0 to ObjPointSet.Count - 1
         Str = ObjPointSet(k).X & " " & ObjPointSet(k).Y
         Stream.WriteLine(Str)
      Next

      ' write blank line after each object
      Stream.WriteLine("")
   Next

   ' close file
   Stream.Close
End Sub
```

Export Text File

```
' Exports selected objects in opened drawing to a text file that
' supports branched objects.
```

```
Sub Main
   Set Windows = Application.WindowSet

   ' fail if there is no opened windows
   If Windows.Count = 0 Then
      Application.MessageBox "No active window"
      Exit Sub
   End If

   Set Window = Windows.ActiveWindow
   Set Component = Window.Component

   ' fail if active component is not a drawing
   If Component.Type <> ComponentDrawing Then
      Application.MessageBox "Active component is not a drawing"
      Exit Sub
   End If

   Set ObjSelection = Component.Selection

   ' fail if there's no selected objects
   If ObjSelection.Count < 1 Then
      Application.MessageBox "No selected objects"
      Exit Sub
   End If

   ' ask for output filename
   FileName = InputBox("File name", "Input")
   Set FSO = CreateObject("Scripting.FileSystemObject")

   ' create file
   Set Stream = FSO.CreateTextFile(FileName, True)

   ' write all selected objects
   For i = 0 to ObjSelection.Count - 1

      ' write object type
      Str = ""
      If ObjSelection.Item(i).Type = ObjectPoint Then
         Str = "Point"
      ElseIf ObjSelection.Item(i).Type = ObjectLine Then
         Str = "Line"
      Else
         Str = "Area"
      End If
      Stream.WriteLine(Str)

      ' write number of branches for lines and areas
      Set ObjBranchSet = ObjSelection(i).Geom.BranchSet
      If ObjSelection(i).Type <> ObjectPoint Then
         Stream.WriteLine(ObjBranchSet.Count)
      End If

      ' write each branch
      For j = 0 to ObjBranchSet.Count - 1
         Set ObjPointSet = ObjBranchSet(j).PointSet
```

```
         If ObjSelection(i).Type <> ObjectPoint Then
            Stream.WriteLine(ObjPointSet.Count)
         End If
         For k = 0 to ObjPointSet.Count - 1
            Str = ObjPointSet(k).X & " " & ObjPointSet(k).Y
            Stream.WriteLine(Str)
         Next
      Next

      ' write blank line after each object
      Stream.WriteLine("")
   Next

   ' close file
   Stream.Close
End Sub
```

Import Release45 Network File

```
' Imports file produced by Release 4.5 "Write Network" solver to a
' new drawing.

Sub Main
   Pi = 3.1415926535897932
   Radius = 10
   CenterX = 10
   CenterY = 15

   ' ask for filename
   FileName = InputBox("File name", "Input")

   Set fso = CreateObject("Scripting.FileSystemObject")
   If fso.FileExists(FileName) Then

      ' open file
      Set Stream = fso.OpenTextFile(FileName, 1, True)

      PtsCount = CInt(Stream.ReadLine)
      LineCount = CInt(Stream.ReadLine)

      ' create new drawing
      Set Drawing = Application.ActiveDocument.NewDrawing("Release45 Network")
      Set Pts = Application.NewPointSet
      Set Brs = Application.NewBranchSet
      Pts.Add Application.NewPoint
      Brs.Add Pts

      ' add point object for each network node
      For i = 0 to PtsCount - 1
         X = CenterX + Radius * sin(2 * Pi / PtsCount * i)
         Y = CenterY + Radius * cos(2 * Pi / PtsCount * i)
         Brs(0).PointSet(0).X = X
         Brs(0).PointSet(0).Y = Y
         Drawing.ObjectSet.Add Application.NewGeom(GeomPoint, Brs)
```

```
      Next

      Set BrsLn = Application.NewBranchSet
      Pts.Add Application.NewPoint
      BrsLn.Add Pts

      ' add line object for each network edge
      For i = 1 to LineCount
         TextLine = Stream.ReadLine
         Tokens = Split(TextLine, " ", 2)
         FirstPt = CInt(Tokens(0))
         SecondPt = CInt(Tokens(1))
         X = CenterX + Radius * sin(2 * Pi / PtsCount * (FirstPt - 1))
         Y = CenterY + Radius * cos(2 * Pi / PtsCount * (FirstPt - 1))
         BrsLn(0).PointSet(0).X = X
         BrsLn(0).PointSet(0).Y = Y
         X = CenterX + Radius * sin(2 * Pi / PtsCount * (SecondPt - 1))
         Y = CenterY + Radius * cos(2 * Pi / PtsCount * (SecondPt - 1))
         BrsLn(0).PointSet(1).X = X
         BrsLn(0).PointSet(1).Y = Y
         Drawing.ObjectSet.Add Application.NewGeom(GeomLine, BrsLn)
      Next

      ' close file
      Stream.Close

      ' open imported drawing
      Drawing.Open
   Else
      Application.MessageBox "File not found"
   End If
End Sub
```

Import Release45 Text File

```
' Imports file produced by Release 4.5 "Write Text File" solver to a
' new drawing.

Sub Main

   ' ask for filename
   FileName = InputBox("File name", "Input")
   Set FSO = CreateObject("Scripting.FileSystemObject")
   If FSO.FileExists(FileName) Then

      ' open file
      Set Stream = FSO.OpenTextFile(FileName, 1)

      ' create new drawing
      Set Drawing = Application.ActiveDocument.NewDrawing("Release45 Text")

      ' read entire file
      Do While Stream.AtEndOfStream <> True
```

```
         ' read object type and number of branches
         ObjHeader = ""
         Do While (Stream.AtEndOfStream <> True And ObjHeader = "")
            ObjHeader = Trim(Stream.ReadLine)
         Loop
         If Stream.AtEndOfStream Then Exit Do

         Header = Split(ObjHeader, " ")
         If UBound(Header) <> 1 Then Exit Do

         ' read points
         Set Brs = Application.NewBranchSet
         ObjType = CInt(Header(0))
         PtsCount = CInt(Header(1))
         Set Pts = Application.NewPointSet
         For i = 0 to PtsCount-1
            TextLine = Stream.ReadLine
            Location = Split(TextLine)
            Set Pt = Application.NewPoint
            Pt.X = CDbl(Location(0))
            Pt.Y = CDbl(Location(1))
            Pts.Add Pt
         Next
         Brs.Add Pts

         ' create object
         If ObjType = 1 Then
            Drawing.ObjectSet.Add Application.NewGeom(GeomPoint, Brs)
         ElseIf ObjType = 2 Then
            Drawing.ObjectSet.Add Application.NewGeom(GeomLine, Brs)
         ElseIf ObjType = 3 Then
            Drawing.ObjectSet.Add Application.NewGeom(GeomArea, Brs)
         End if
      Loop

      ' close file
      Stream.Close

      ' open imported drawing
      Drawing.Open
   Else
      Application.MessageBox "File not found"
   End if
End Sub
```

Import Text File

```
' Imports file produced by "Export Text File" solver to a new drawing.

Sub Main

   ' ask for filename
   FileName = InputBox("File name", "Input")
   Set FSO = CreateObject("Scripting.FileSystemObject")
```

```
If FSO.FileExists(FileName) Then

   ' open file
   Set Stream = FSO.OpenTextFile(FileName, 1)

   ' create new drawing
   Set Drawing = Application.ActiveDocument.NewDrawing("Release45 Text")

   ' read entire file
   Do While Stream.AtEndOfStream <> True

      ' read object type and number of branches
      ObjType = ""
      Do While (Stream.AtEndOfStream <> True AND ObjType = "")
         ObjType = Stream.ReadLine
      Loop
      If Stream.AtEndOfStream Then Exit Do
      If ObjType = "Point" Then
         BrsCount = 1
      Else
         BrsCount = CInt(Stream.ReadLine)
      End If

      ' read or synthesize branches
      Set Brs = Application.NewBranchSet
      For i = 0 to BrsCount - 1
         If ObjType = "Point" Then
            PtsCount = 1
         Else
            PtsCount = CInt(Stream.ReadLine)
         End If
         Set Pts = Application.NewPointSet
         for j = 0 to PtsCount - 1
            TextLine = Stream.ReadLine
            Location = Split(TextLine)
            Set Pt = Application.NewPoint
            Pt.X = CDbl(Location(0))
            Pt.Y = CDbl(Location(1))
            Pts.Add Pt
         Next
         Brs.Add Pts

         ' create object
         If ObjType = "Point" Then
            Drawing.ObjectSet.Add Application.NewGeom(GeomPoint, Brs)
         ElseIf ObjType = "Line" Then
            Drawing.ObjectSet.Add Application.NewGeom(GeomLine, Brs)
         ElseIf ObjType = "Area" Then
            Drawing.ObjectSet.Add Application.NewGeom(GeomArea, Brs)
         End if
      Next
   Loop

   ' close file
   Stream.Close

   ' open imported drawing
```

```
      Drawing.Open
   Else
      Application.MessageBox "File not found"
   End if
End Sub
```

Tranfer Selection
(select Pixels under selected objects)

Select Pixels under Selected Objects

```
' Modifies selection within "Vatican Image" image to match that in
' "Vatican" drawing.

Sub Main
   Set Components = Document.ComponentSet

   ' obtain image and drawing
   Set Drawing = Components(Components.ItemByName("Vatican"))
   Set Image = Components(Components.ItemByName("Vatican Image"))

   ' set up coordinate converter
   Set SystemImage = Image.CoordinateSystem
   Set SystemDrawing = Drawing.CoordinateSystem
   Set Converter = Application.NewCoordinateConverter
   Converter.Prepare SystemImage, SystemDrawing

   Set Point = Application.NewPoint
   Set Pixels = Image.PixelSet
   Set SelectedObjects = Drawing.Selection

   ' batch updates for performance
   Document.BatchUpdates = True

   ' traverse image pixels modifying selection state as necessary
   For Index = 0 To Pixels.Count-1
      If Index Mod 100 = 0 Then
         Application.StatusText = CStr(Index) + " of " + CStr(Pixels.Count) + "
pixels."
      End If

      Set Pixel = Pixels(Index)

      ' deselect pixel
      Pixel.Mask = Pixel.Mask And Not 1

      ' send pixel coordinates to geometric entity taking care of inverse Y direction
      Point.X = Pixel.X
      Point.Y = Image.Height - Pixel.Y - 1
      Set PointGeom = Application.NewGeom(GeomPoint, Point)

      ' convert pixel coordinates to drawing
      Converter.Convert PointGeom
```

```
      ' loop through selected objects within drawing to check if there's one that
contains pixel
      For IndexObj = 0 To SelectedObjects.Count-1
         Set Object = SelectedObjects.Item(IndexObj)
         If Object.Geom.CheckContains(PointGeom) Then
            Pixel.Mask = Pixel.Mask Or 1
            Exit For
         End If
      Next
   Next

   ' process batched updates
   Document.BatchUpdates = False

   Application.StatusText = ""
End Sub
```

Select Pixels with 50 Meters of Selected Objects

```
' Modifies selection within "Vatican Image" image so that each pixel
' within 50 meters of any of selected objects in "Vatican" drawing is
' also selected and all other pixels are unselected.

Sub Main
   Set Components = Document.ComponentSet

   ' obtain image and drawing
   Set Drawing = Components(Components.ItemByName("Vatican"))
   Set Image = Components(Components.ItemByName("Vatican Image"))

   ' set up coordinate converter
   Set SystemImage = Image.CoordinateSystem
   Set SystemDrawing = Drawing.CoordinateSystem
   Set Converter = Application.NewCoordinateConverter
   Converter.Prepare SystemImage, SystemDrawing

   Set Point = Application.NewPoint
   Set Pixels = Image.PixelSet
   Set SelectedObjects = Drawing.Selection

   ' batch updates for performance
   Document.BatchUpdates = True

   ' traverse image pixels modifying selection state as necessary
   For Index = 0 To Pixels.Count-1
      If Index Mod 100 = 0 Then
         Application.StatusText = CStr(Index) + " of " + CStr(Pixels.Count) + "
pixels."
      End If

      Set Pixel = Pixels(Index)

      ' deselect pixel
```

```
      Pixel.Mask = Pixel.Mask And Not 1

      ' send pixel coordinates to geometric entity taking care of inverse Y direction
      Point.X = Pixel.X
      Point.Y = Image.Height - Pixel.Y - 1
      Set PointGeom = Application.NewGeom(GeomPoint, Point)

      ' convert pixel coordinates to drawing
      Converter.Convert PointGeom

      ' loop through selected objects within drawing to check if there's one that
contains pixel
      For IndexObj = 0 To SelectedObjects.Count-1
         Set Object = SelectedObjects.Item(IndexObj)
         If Object.Geom.Distance(PointGeom) < 50 Then
            Pixel.Mask = Pixel.Mask Or 1
            Exit For
         End If
      Next
   Next

   ' process batched updates
   Document.BatchUpdates = False

   Application.StatusText = ""
End Sub
```

Capturing Sreenshots of the Active Component

---------- Example description:

This example add-in demonstrates how to make quick screenshots of the active component (analogous to the output of the MakeImage command in the "entire component" mode).

The example illustrates:

- how to obtain the currently opened component,
- how to screenshot a component into a new image of desired size,
- how to place a component into a folder creating one if necessary.

---------- To deploy the example:

1. Copy the files in the Add folder into a new folder under Config.

   Open the Screenshots.xml file (with Notepad) and correct the paths to the script files given in lines 8, 13 and 18.

2. Restart Manifold.

```
----------  SCREENSHOT XML ----------
```

```
- <xml>
- <addin>
  <name>Screenshots</name>
  <copyright>CDA International, 2003</copyright>
  <description>Takes quick screenshots of currently opened component at different
resolutions.</description>
- <command>
  <name>Screenshot (400x300)</name>
  <path>Screenshots\ScreenshotSmall.txt</path>
  <status>Takes a small screenshot of currently opened component.</status>
  </command>
- <command>
  <name>Screenshot (800x600)</name>
  <path>Screenshots\ScreenshotMedium.txt</path>
  <status>Takes a screenshot of currently opened component.</status>
  </command>
- <command>
  <name>Screenshot (2400x1800)</name>
  <path>Screenshots\ScreenshotLarge.txt</path>
  <status>Takes a large screenshot of currently opened component.</status>
  </command>
  </addin>
  </xml>
```

---------- LARGE SCREENSHOT ----------

```
' ************************************************************************

Sub Screenshot(x, y)

   ' obtain context component and ensure it is drawing, image, labels, map or surface
   Set windowSet = Application.WindowSet
   If  windowSet.Count <= 0 Then
      Application.MessageBox "No opened windows."
      Exit Sub
   End If
   Set window = windowSet.ActiveWindow
   Set component = window.Component
   If  component.Type <> ComponentDrawing And _
      component.Type <> ComponentImage And _
      component.Type <> ComponentLabels And _
      component.Type <> ComponentMap And _
      component.Type <> ComponentSurface Then
      Application.MessageBox "Active window should contain drawing, image, labels, map
or surface."
      Exit Sub
   End If

   ' render context component to image and place image into a folder named
"Screenshots"
   Set document = Application.ActiveDocument
   If  document.ReadOnly Then
      Application.MessageBox "Can't create new image component.  Document is read-
only."
```

```
      Exit Sub
   End If

   folderName = "Screenshots"
   renderName = component.Name & " (" & x & "x" & y & ") " & Date & " " & Time

   ' render context component to image
   Call component.RenderTo(renderName, x, y, True)

   ' locate rendered image and move it into a folder named "Screenshots"
   renderIndex = document.ComponentSet.ItemByName(renderName)
   If renderIndex >= 0 Then
      Set render = document.ComponentSet.Item(renderIndex)
      If  render.Type = ComponentImage Then
         folderIndex = document.ComponentSet.ItemByName(folderName)
         If  folderIndex < 0 Then
            Set folder = document.NewFolder(folderName)
         Else
            Set folder = document.ComponentSet.Item(folderIndex)
         End If
         If  folder.Type = ComponentFolder Then
            render.Folder = folder
         End If
      End If
   End If
End Sub

' ****************************************************************************

Sub Main
   Call Screenshot(2400, 1800)
End Sub

' ****************************************************************************
```

---------- MEDIUM SCREENSHOT ----------

```
' ****************************************************************************

Sub Screenshot(x, y)

   ' obtain context component and ensure it is drawing, image, labels, map or surface
   Set windowSet = Application.WindowSet
   If  windowSet.Count <= 0 Then
      Application.MessageBox "No opened windows."
      Exit Sub
   End If
   Set window = windowSet.ActiveWindow
   Set component = window.Component
   If  component.Type <> ComponentDrawing And _
      component.Type <> ComponentImage And _
      component.Type <> ComponentLabels And _
```

```
      component.Type <> ComponentMap And _
      component.Type <> ComponentSurface Then
      Application.MessageBox "Active window should contain drawing, image, labels, map
or surface."
      Exit Sub
   End If

   ' render context component to image and place image into a folder named
"Screenshots"
   Set document = Application.ActiveDocument
   If  document.ReadOnly Then
      Application.MessageBox "Can't create new image component.  Document is read-
only."
      Exit Sub
   End If

   folderName = "Screenshots"
   renderName = component.Name & " (" & x & "x" & y & ") " & Date & " " & Time

   ' render context component to image
   Call component.RenderTo(renderName, x, y, True)

   ' locate rendered image and move it into a folder named "Screenshots"
   renderIndex = document.ComponentSet.ItemByName(renderName)
   If renderIndex >= 0 Then
      Set render = document.ComponentSet.Item(renderIndex)
      If  render.Type = ComponentImage Then
         folderIndex = document.ComponentSet.ItemByName(folderName)
         If  folderIndex < 0 Then
            Set folder = document.NewFolder(folderName)
         Else
            Set folder = document.ComponentSet.Item(folderIndex)
         End If
         If  folder.Type = ComponentFolder Then
            render.Folder = folder
         End If
      End If
   End If
End Sub

' ************************************************************************

Sub Main
   Call Screenshot(800, 600)
End Sub

' ************************************************************************
```

---------- SMALL SCREENSHOT ----------

```
' ************************************************************************

Sub Screenshot(x, y)
```

```
   ' obtain context component and ensure it is drawing, image, labels, map or surface
   Set windowSet = Application.WindowSet
   If  windowSet.Count <= 0 Then
      Application.MessageBox "No opened windows."
      Exit Sub
   End If
   Set window = windowSet.ActiveWindow
   Set component = window.Component
   If  component.Type <> ComponentDrawing And _
      component.Type <> ComponentImage And _
      component.Type <> ComponentLabels And _
      component.Type <> ComponentMap And _
      component.Type <> ComponentSurface Then
      Application.MessageBox "Active window should contain drawing, image, labels, map
or surface."
      Exit Sub
   End If

   ' render context component to image and place image into a folder named
"Screenshots"
   Set document = Application.ActiveDocument
   If  document.ReadOnly Then
      Application.MessageBox "Can't create new image component.  Document is read-
only."
      Exit Sub
   End If

   folderName = "Screenshots"
   renderName = component.Name & " (" & x & "x" & y & ") " & Date & " " & Time

   ' render context component to image
   Call component.RenderTo(renderName, x, y, True)

   ' locate rendered image and move it into a folder named "Screenshots"
   renderIndex = document.ComponentSet.ItemByName(renderName)
   If renderIndex >= 0 Then
      Set render = document.ComponentSet.Item(renderIndex)
      If  render.Type = ComponentImage Then
         folderIndex = document.ComponentSet.ItemByName(folderName)
         If  folderIndex < 0 Then
            Set folder = document.NewFolder(folderName)
         Else
            Set folder = document.ComponentSet.Item(folderIndex)
         End If
         If  folder.Type = ComponentFolder Then
            render.Folder = folder
         End If
      End If
   End If
End Sub

' ********************************************************************

Sub Main
   Call Screenshot(400, 300)
End Sub
```

```
' ************************************************************************
```

Active Columns

Active Columns Cannot be Added Programmatically – Work Around

http://69.17.46.171/Site/Thread.aspx?id=27504&ti=632939834854070000

jonno on 9/13/2006 7:44 AM (#27505)
Hi All,

Having pored through the documentation and this forum, would I be right in stating that we cannot add an Active Column programmatically?

Jonathan

adamw on 9/16/2006 6:04 AM (#27664)
Yes. There is no way to add an active column using a script. A common workaround is to keep a dummy active column and alter its Function property as well as the contents of the relevant script component.

Create Active Column (script component must be previously attached to the table)

Date: Fri 02/17/2006 2:17 AM
From: Adam Wachowski [adamw@manifold.net]
To: Manifold List
Subject: Re: [Manifold-l] Can Active Column Creation Be Scripted
http://lists.directionsmag.com/discussion/read.php?f=29&i=41596&t=41572

> Does anyone know if active column creation can be done via script? If
> so can the script for the column(s) also be created this way?

Yes:

```
Sub Main
  Set tbl = Application.ActiveDocument.ComponentSet("Drawing Table")
  Set scr = Application.ActiveDocument.ComponentSet("Drawing Table Script")

   ' add new function
 colScript = "$ Function MyFunc $  MyFunc=CInt(Record.Data(""ID""))+1 " +_
   "$ End Function $"
 scr.Text = scr.Text + Replace(colScript, "$", Chr(13) + Chr(10))

 ' add new column
 Set col = tbl.ColumnSet.NewColumn()
 col.Name = "NewColumn"
 col.Type = ColumnTypeInt32U
 col.Function = "MyFunc"
```

```
 col.ComputationMode = ComputationModeManual
 tbl.ColumnSet.Add col
End Sub
```

You have to have a script component attached to the table already. The easiest way to achieve this is to create a new active column and then delete it (and optionally remove all code from the script component). I think you can not yet do this programmatically.

Adam Wachowski

Calculating Spherical Distances and Areas

L. Ketch
This is a two part topic.
1. a note I sent to Manifold©-L and a reply from Adam
2. some information gleaned from the old GeoReference site with an Active Column solution for calculating spherical area and distance.

http://forum.manifold.net/Site/Thread.aspx?id=14517&ti=632622901030000000

adamw on 9/7/2005 8:48 AM (#14563)
I modified it to determine spherical distance / speed. It calculates the same values that Mapinfo© would give.

Sent: Mon 01/23/2006 4:37 PM
To: 'Manifold-L@lists.directionsmag.com'
From: Lorne [lketch@hfx.eastlink.ca]
Subject: Spherical Calculation of Distance and Area
http://lists.directionsmag.com/discussion/read.php?f=29&i=41388&t=41388

I am trying to gain some insight into how Manifold handles distance and area calculations. My Drawing consists of storm tracks that can cover much of the North Atlantic Ocean. A single storm consists of multiple lines segments with each segment representing a 6-hour portion of the track. Tabular data such as pressure, maximum wind speed, etc are tied to each track segment. Track segments for a given event can be grouped by a storm ID.

I would like to be able to calculate the length of each 6-hour segment in nautical miles. That figure will be divided by 6 to get a 6-hour speed in knots.

When I was using Mapinfo, this calculation could be done using an Update SQL statement like that shown below. That string returns the same numbers regardless of the projection or lack of - i.e. it works for Latitude/Longitude maps as well.

```
Update Tracks Set SystemSpeed_kts = Round(ObjectLen(obj, "nmi") / 6.0, 0.1)
```

Manifold requires a projected map or an SQL string that casts the Geoms to a projection (I have tried both approaches). Something like:

```
Select Round( (Length (  Project (  [Tracks_2004 Drawing].[Geom (I)],
COORDSYS("Equidistant Conic") ), "nm" ) / 6) ,1) as Speed
from [Tracks Drawing]
```

I am finding that the numbers change depending upon which coordinate system is used. Further, none of the numbers match the old values calculated in Mapinfo. Higher speeds can show considerable variation from the MI values.

Questions:
How is Manifold© actually calculating these values? Does the Length function use great circle distances? Would this have something to do with a coordinate system being applied to a table of objects that span a large geographic area and each object using that default table system rather than one centered on the object?

Thanks
Lorne

Sent: Tue 01/24/2006 10:38 AM
To: Manifold List
From: Adam Wachowski [adamw@manifold.net]
Subject: RE: [Manifold-l] Spherical Calculation of Distance and Area
http://lists.directionsmag.com/discussion/read.php?f=29&i=41405&t=41405

> How is Manifold actually calculating these values?

It uses Euclidean formulae in the projection of the geom. Thus, converting the coordinates of a geom to another projection changes the length and area of the geom as well.

> Does the Length function use great circle distances?

No.

We have a wishlist item to allow computing the length or area of a geom over the ellipsoid in a projection-neutral way, which seems to be similar to what is being done by MapInfo. This might appear in a future update.

Adam Wachowski

Anne
http://www.georeference.org/Forums/tabid/71/forumid/1/tpage/1/view/topic/postid/12284/Default.aspx#12472

09/05/2005 5:33 PM
Hello All,
I am putting together a polygon layer of covenanted land areas in New Zealand. Our database system records the area (in Hectares) that the land surveyor calculated when each covenant was

surveyed. I am comparing this reported area to the area of the digital polygons as calculated in Manifold. There are over 2000 covenants which range from < 1 ha up to 6500 ha but the median size is 6.4 ha.

I have two concerns:

1. The intrinsic Area(I) calculation is consistently above the reported area for each covenant AND when compared to area calculated in ArcGIS is consistently above that as well. Out of 1715 covenants, the Manifold Area calculation is greater than the Arc calculation of the same polygons by an average of 7.6% of the expected area (with a std dev of 0.2, so it is pretty consistent).

2. The intrinsic Area(I) field should report area in METRES squared but actually reports it in DECIMETRES squared, ie. I have to divide by 100,000 instead of 10,000 in order to convert to hectares.

I am using Manifold 6.5 and the drawing is projected into New Zealand Map Grid (1949 datum). Can anyone shed any light on these issues?
Thanks,

09/07/2005 8:48 AM

> amb Posted:09/05/2005 2:33 PM
> The intrinsic Area(I) calculation is consistently above the reported area for each covenant AND when compared to area calculated in ArcGIS is consistently above that as well.

The Area (I) column contains the Euclidean area in whatever projection the drawing is in. To compute the real geographic area, use an active column, eg:

```
Function A
    Set objs = Table.Owner.ObjectSet
    Set obj = objs(objs.ItemByID(Record.Data("ID")))
    A = obj.Geom.Area / (1000*1000) ' sq kilometres
End Function
```

NOTE:
Code below is a modification of Adam's to calculate speed in nautical miles for the North Atlantic HURDAT tropical cyclone archive – time step is 6 hours.
http://www.nhc.noaa.gov/pastall.shtml
(L. Ketch)

```
Function Spherical_Speed
     ' Table is Azimuthal Equidistant World Geodetic 1984 (WGS84)
   Set objs = Table.Owner.ObjectSet
   Set obj =  objs(objs.ItemByID(Record.Data("ID")))
       ' convert metres to Nautical Miles
   Length_nm = obj.Geom.Length * 0.000539594075
   Spherical_Speed =  Round(Length_nm / 6.0, 1)
End Function
```

amb Posted:09/05/2005 2:33 PM

The intrinsic Area(I) field should report area in METRES squared but actually reports it in DECIMETRES squared,

Make sure the drawing is georegistered correctly. It sounds like the local scale parameters are ten times what they should be.

Adam Wachowski

09/13/2005 12:25 AM
Thanks Adam,

That script seemed to do the trick. All areas look good now.
Haven't solved the unit problem though because the parameters look ok for the NZMG projection.

Cheers, Anne

09/14/2005 5:34 AM
Euclidean area is the literal area value you would get when determining the area using the coordinate system of the data. Imagine drawing a shape on a lat/lon plot - if you use the literal x and y values in that plot it won't be the same as if that area were drawn on the Earth's spheroid. Try drawing a "Geographic Circle" on a lat/lon map, and other projections and you'll see it clearly.

All intrinsic values except for Bearing (I) are calculated this way - so the intrinsic values are dependent upon the projection you are using.

I'm not sure why the Bearings don't behave as the others do, but I'm sure it's to do with the fact that bearings are a little more screwily dependent on the coordinate system, and how you are defining what is meant by direction.

09/14/2005 9:26 AM
Another term for it is **spherical excess** - you get it both in areas and angles. The contained angles of a triangle drawn on a planar surface will total 180 degrees. When drawn on a spheroid they will total more - how much more depends on the ratio between the size of triangle and the size of the sphere. If you can imagine the skin of the sphere delineated by the triangle it would appear tent-like and therefore has more area.

David M Brubacher OLS OLIP

Two or More Active Columns in One Table

http://69.17.46.171/Site/Thread.aspx?id=20942&ti=632828332475870000

Elvir at 5/10/2006 2:27 AM (#20943)
Is it possible to have two Active columns in one Table?

KlausDE at 5/10/2006 4:47 AM (#20946)
The trick is there is only one script for one table with a function for each of the active columns. So for the second active column you get the script with the function for the 1st structure and have to add the 2nd function from scratch.

Create a Column Counter (Sequenced Number) using an Active Column

http://69.17.46.171/Site/Thread.aspx?id=111&ti=632053080800000000

adamw on 11/22/2003 7:29 AM (#134)
Here is an example that shows how to create a counter column:

Create a new MAP file. Create a new table with Name (text) and Key (integer) columns. Open table. Add active column named KeyService, set function name to KeyService, column type to integer, and column computation mode to "on demand." Set script text to:

```
Function KeyService
   If Record.Data("Key") = 0 Then
      Record.Data("Key") = RecordSet.Maximum("Key")(0).Data("Key") + 1
   End If

   KeyService = Record.Data("Key")
End Function
```

L. Ketch
V7 allows a much faster running solution.

```
Function KeyService
   KeyService = Record.Index
End Function
```

Switch to table. Click the cell under the Name column. Enter some text. Click Enter. Click Shift-Enter to append the new record. Add more records. The KeyService function will compute unique IDs and put them into both the KeyService and Key columns.

You might want to set the computation mode of the KeyService column to "on user request," add several new records (which will have a default value of 0 in the Key column), then right-click the KeyService column and invoke Recompute. This will compute the IDs for the newly added records and save them in both Key and KeyService columns.

The code above will work fine for small tables, but will probably be too slow for large amounts of (new) records. Having said that, it is easy to modify the code so that it either looks up the unused ID in a comment component, or in Table.Description. One could also modify the code so that the IDs are generated by the COM object. Given the size of the DLL containing the COM object is not overly large, this could actually be faster than any other method.

adamw on 11/22/2003 1:08 PM (#147)

> *Since it is a key, we don't need them sequential and we really don't want them to be re-used.*

One more idea: if keys are only necessary for maintaining identity, why not use GUIDs? This will automatically ensure the uniqueness (within the component, within the MAP file, and within all other MAP files for that matter).

> *I don't believe the "natural language" type of query builder is what would be desirable but rather one like that provided by Access and VS.Net ...*

Point taken.

dmbrubac on 11/22/2003 1:58 PM (#151)

GUIDs are good. Can you create them natively with manifold?

adamw on 11/22/2003 3:31 PM (#153)

No, but you can easily generate GUIDs in VB.NET with Guid.NewGuid(). Then you will consume the GUIDs in a Manifold script connecting to a .NET module through COM.

rheitzman on 11/23/2003 1:24 PM (#162)

> *No, but you can easily generate GUIDs in VB.NET with Guid.NewGuid(). Then you will consume the GUIDs in a Manifold script connecting to a .NET module through COM.*

Adam - could you post a simple example of how to access the .Net Framework via COM from VBScript?

Are we talking passing code in text form or executing a "compiled" application?

Thanks!

adamw on 11/24/2003 6:22 AM (#179)

Here is a short example that uses VB.NET 2003 to develop a .NET object and calls this object from Manifold script through COM.

Start VB.NET 2003. Invoke File - New Project, select Visual Basic Projects as a project type, select Class Library as a template. Set project name to GuidGenerator. Click OK.

Change the contents of the Class1.VB to:

```
Imports System.Runtime.InteropServices

    ' Interface definition
Public Interface IGuidGenerator
   ReadOnly Property GuidCount As Integer
   Function NewGuid() As String
End Interface

' Implementation code
<ClassInterface(ClassInterfaceType.None), ProgId("Test.GuidGenerator")> _

Public Class GuidGenerator

Implements IGuidGenerator

' Number of generated GUIDs

Private Generated As Integer

' Create new GUID generator

Public Sub GuidGenerator()
   Generated = 0
End Sub

' Obtain number of generated GUIDs

Public ReadOnly Property GuidCount As Integer Implements IGuidGenerator.GuidCount
   Get
      Return Generated
   End Get
End Property

' Generate new GUID

Public Function NewGuid() As String Implements IGuidGenerator.NewGuid
   Generated = Generated + 1
   Return Guid.NewGuid().ToString()
End Function
```

```
End Class
```

Right-click the project item (not the solution item) in the Solution Explorer. Select All Configurations as a configuration. Navigate to Configuration Properties - Build. Check "Register for COM Interop." Click OK.

Invoke Build - Build Solution to build the .NET class library and register it as a COM module.

Start Manifold. Create a new script component. Set script code to:

```
Sub Main
   Set generator = CreateObject("Test.GuidGenerator")
   Application.MessageBox generator.NewGuid()
   Application.MessageBox generator.NewGuid()
   Application.MessageBox generator.GuidCount
End Sub
```

Run the script. This should display two GUID strings, and then the number 2.

Calculate the Distance in Miles from Given Point to All Geoms

From: adamw@manifold.net
Sent: September 19, 2006 9:52 AM
To: manifold-l@lists.directionsmag.com
Subject: RE: [Manifold-l] Distance calculations
http://lists.directionsmag.com/discussion/read.php?f=29&i=42556&t=42556

> I am getting a type Mismatch error on NewPointLatLon when I set up the
> following function in an ActiveColumn:
>
> Function DStudio
>   DStudio = distance([Geom (I)], NewPointLatLon(-77.0770,38.9657), "mi")
> End Function

You are using query functions in a script.

Try this:

```
Function DStudio
   Set p = Application.NewGeomFromTextWKT("POINT(-77.0770 38.9657)")
   DStudio = Record.Object.Geom.Distance(p, 0.001) / 1609    ' in miles
End Function
```

Adam Wachowski

L. Ketch: The Geom Method above is defined as:

Number Distance(Geom geom, Number epsilon)
*Returns distance to given entity measured over an **ellipsoid**. Returned value is in meters.*

The return value is in metres and Adam is using 1609 to convert to miles. Adam has set epsilon to 0.001metre.

> *"Within the Manifold programming and mathematics team the location precision parameter is called the epsilon. Some documentation and notes within sample source code may refer to "the epsilon" used for a given action. This is the location precision distance referred to by a more mathematical name."*

Active Column to Reference the "Previous" Record in a Table

http://forum.manifold.net/Site/Thread.aspx?id=31584&ti=633011039604370000

spoedniek on 12/7/2006 12:16 PM (#31704)
Not sure how much this will help, but in jscript I access the previous record something like this:

```
' JScript
// Prev: An initial value declared and initialised outside the
//       function which retains its value through iterations.
Var Prev = 0;

Function getPrevious()
{
  Var valueToBeReturned = Prev;
  Prev = Record.Data( "Column" );
  Return valueToBeReturned;
}
```

The *Prev* variable is declared outside the function (and is therefore global) and retains its value through the iterations. I haven't experimented much with this as I've only needed it a couple of times for sequences, so I'm not sure what else is possible with the active column code.

Henry

Analyzer Object

(some of these examples can be found elsewhere in this document)

Select Touch All Objects in All Map Layers that Touch a Bounding Object

sitesatlas at 6/1/2006 2:40 PM (#22026)

Thank you so much. I just made a couple little changes and it worked like a charm:

```
Sub Main
Set ZActiveDoc = Application.ActiveDocument
Set ZCompSet = ZActiveDoc.ComponentSet
Set ZBoundBox = ZCompSet.Item("Bounding Box")
Set ZAnalyzer = ZActiveDoc.NewAnalyzer
for each layer1 in ZCompSet.Item("Map").LayerSet
ZAnalyzer.SelectTouching layer1.Component, layer1.Component.ObjectSet,
ZBoundBox.ObjectSet
next
End Sub
```

Explode Polyline

http://69.17.46.171/Site/Thread.aspx?id=27362&ti=632937183439630000

Art Lembo Provided a Script Version using the **Analyzer Object**

artlembo on 9/10/2006 6:16 PM (#27363)

```
Sub Main
  Set Drwg = Application.ActiveDocument.ComponentSet.Item("Drawing")
  Set Analyzer = Application.ActiveDocument.NewAnalyzer()
  Set Objs = Drwg.ObjectSet
  Analyzer.Explode Drwg, Objs
End Sub
```

Compute Shortest Distance from Point A to a Road

http://69.17.46.171/Site/Thread.aspx?id=19151&ti=632902448063970000

artlembo on 2/23/2006 11:58 AM (#19160)

This was a project where we had to compute the distances between a chicken coop, and all locations where chicken manure was spread. And, rather than "as the crow flies" we had to do it "as the tractor drives". We then put the distances in a table.

Now, why would we do such a thing? Well, chicken manure is really potent stuff, and adds lots of nitrogen into the ground. Most farmers when spreading it usually dump it off as close to the

coop as possible, so you should see higher nitrogen values there. We were attempting to correlate N with distance from the coop.

```
Sub Main
   Set theActiveDoc = Application.ActiveDocument
   Set theCompSet = Document.ComponentSet
   Set Fields = theCompSet.Item("chickens")
   Set Roads = theCompSet.Item("roads")
   Set VC = theCompSet.Item("Chicken Coop")
   Set Analyz = theActiveDoc.NewAnalyzer()
   Set theTable = theCompSet.Item("Dists")
   Set VCObj = VC.ObjectSet.Item(0)
   Set PortObjSet = Fields.ObjectSet

   i = 0
   for each rec in Fields.ObjectSet
     FieldName = Fields.OwnedTable.RecordSet.Item(i).Data("ID2")
     i = i + 1
     Set Spath = Analyz.SelectShortestPath(Roads, Roads.ObjectSet, rec, VCObj)
     if Spath.Count > 0 then
      thedist = Spath.GeomSet.JoinLines.Item(0).Length
         ' Application.Messagebox FieldName & " " & thedist
      theTable.RecordSet.AddNew()
      theTable.RecordSet.LastAdded.Data("NAME") = FieldName
      theTable.RecordSet.LastAdded.Data("dist") = thedist

     end if
   Next
End Sub
```

Iterating over the Analyzer ObjectSet

adamw on 8/3/2006 6:29 AM (#25440)

Analyzer.SelectShortestPath returns an ObjectSet, which you can iterate over directly using either For Each or For:

'VBScript

```
  ' method 1
For Each o In spath
  t = o.Record.Data("Length (I)") * o.Record.Data("Speed")
  ...
Next

  ' method 2
For i = 0 To spath.Count - 1
  Set o = spath(i)
  t = o.Record.Data("Length (I)") * o.Record.Data("Speed")
  ...
Next
```

COM Object Model – External Programming Environments NET, VBA, etc.

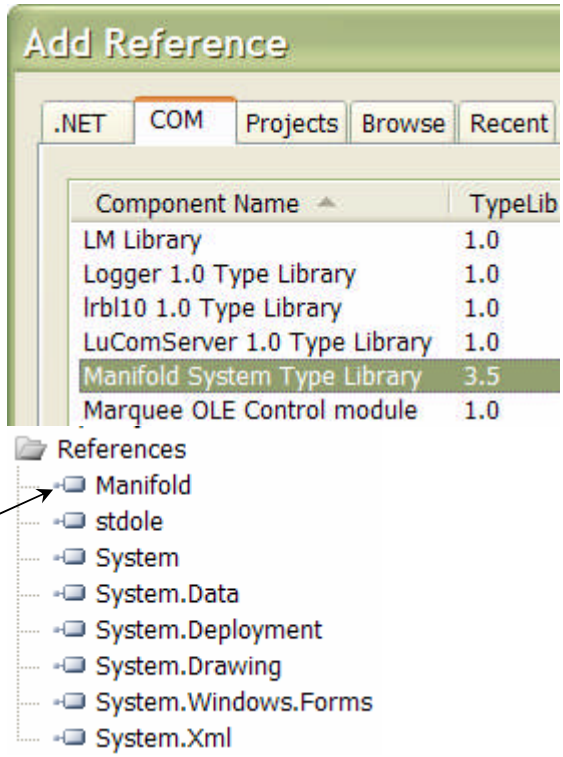
Basic Setup Operations to Access the Manifold.Interop.dll Assembly

L. Ketch

Typical project setup actions:

Setting a reference to Manifold.Interop

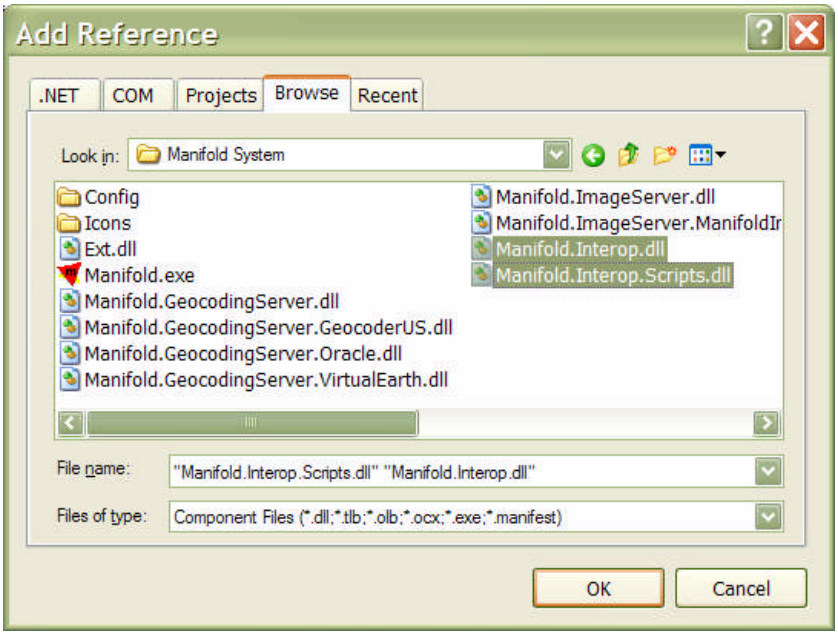
- Menu | Project | Add Reference
- Select the COM tab
- Select "Manifold© System Type Library" (V7x name)



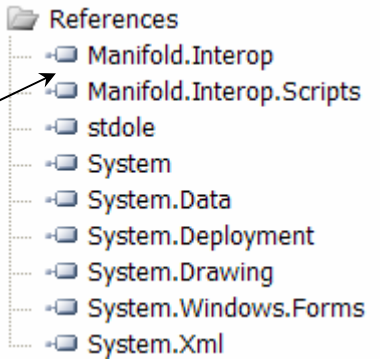
The "Manifold" reference will now show under References.

Setting a references to:
Manifold.Interop
and
Manifold.Interop.Scripts

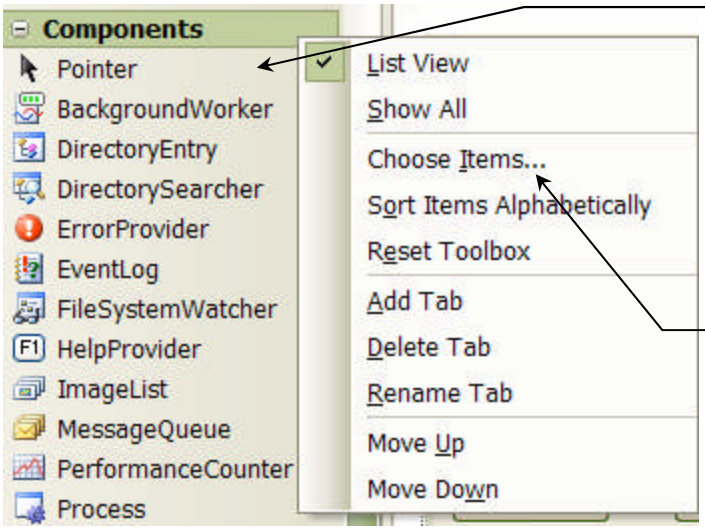
- Click the Browse tab of the Add Reference dialog.
- Browse to the Manifold© installation directory.
- Select the two DLL's.
- Click Ok



References will now show
Manifold.Interop
and
Manifold.Interop.Scripts



Adding the Manifold MapControl to the Toolbox.

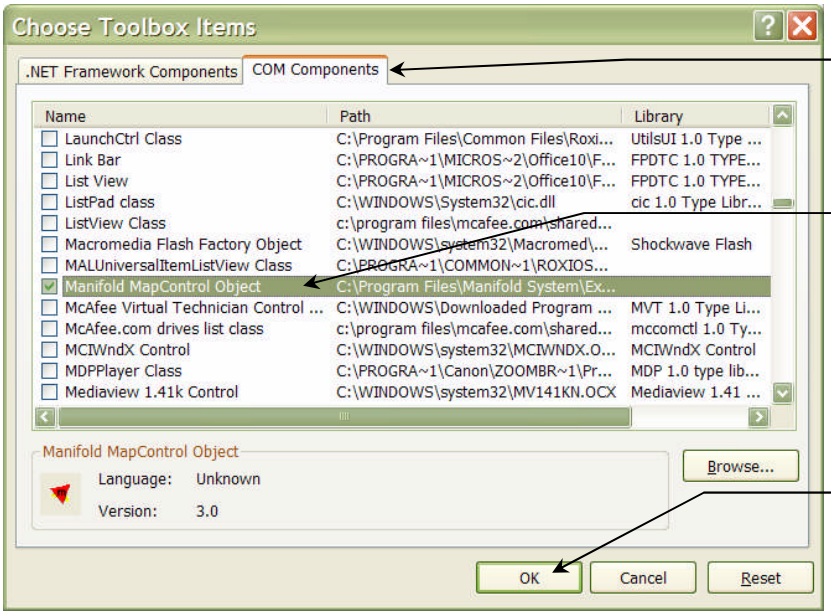


If the Manifold© MapControl is not listed in the toolbox, then:

- Right click in an open area of the Toolbox.
- A popup selection box opens.

Click the
Choose Items…
Item.

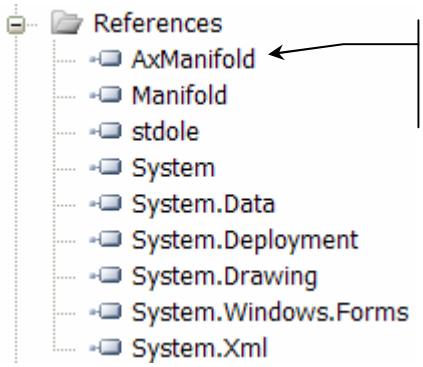
The menu below opens.



Select the
COM Components
Tab

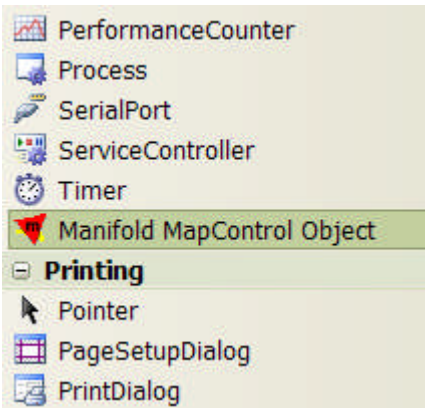
Select the
Manifold© MapControl Object
Tab (click the checkbox on)

Click **OK**



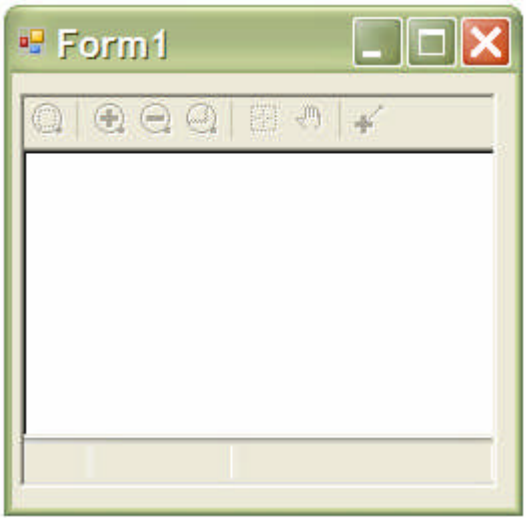
References will now show the AxManifold Control

Adding a Map Component to a Form:



The Manifold control should now be available in the Toolbox as shown to the left.

It can be added to a Form like any of the standard controls.



How to use the Manifold© Object Model in VB 2005 (project initial setup)

http://forum.manifold.net/Site/Thread.aspx?id=31261&ti=633018805024700000

dmcullen on 11/30/2006 2:04 PM (#31262)

Does anyone know of a good way to understand how to use the Manifold Object Model in Visual Basic 2005? I have watched the GIS Advisors VB Script training video, which is very helpful for scripting in Manifold, but I am struggling with implementing in VB 2005. How do I do something like Zoom to Selection? It seems so simple in Manifold, but I must not be using the objects correctly in VB 2005. I will start another thread with that question.
--dmcullen

gxdata on 12/4/2006 8:07 AM (#31456)

I'm glad your question mentions Visual Basic 2005, since it implies that you are working within the Visual Studio (VS2005) IDE (development environment).

It's MUCH harder when trying to work out the Manifold object model when one is scripting (with VBScript) within the Manifold user interface.

However - There is an object model diagram that can be downloaded.

The User Guide has sections on scripting and programming. They are brief, but useful. Personally, I would suggest that they would be better separated into another document or Programmers' Guide, along with some quite simple help information, but the information is available - albeit sparse and stark.

Within Visual Studio:

Once you have set a reference in VS2005 to Manifold.Interop, don't you have IntelliSense kicking in to help you, like other objects referenced within the VS IDE? That is very helpful.

It is possible to use the Object Browser panel within Visual Studio, and have a look at the object model that way.

If you understand the rudiments of working within Visual Studio, it really is worthwhile trying some fairly simple things there, and then (if it is a better way to achieve what you want within the Manifold project (.MAP) environment, without an external application), to take the principles you have learned and do the same lessons or exercises with VBScripts.

It would probably be helpful to some people to see much the same simple operations done in both ways. It would not be difficult to set up simple tutorials that did that.

We're not all trying to be gun programmers, after all.

KlausDE on 12/16/2006 3:41 PM (#32323)

1. Add a Reference to Manifold.Interop in VS2005
   Menu → Project → Add Reference → COM → Manifold System Type Library → OK

2. Create a Form
   a) Add the Manifold ActiveX from the Toolbox to the Form.
      The ActiveX Control is named "Manifold MapControl Object".
      (In the Object Browser and in many threads in this forum you will find it named "AxComponentControl" in AxManifold.Interop
   b) Set properties Filename and Component to the path to an existing project and a component name existing in this project.

Creating a Custom Add-in Pane (from Manifold© Help)

Add-ins can be used to create custom panes. This example demonstrates how to create a custom pane using Visual Basic 2005 Express.

1. Launch Visual Basic 2005 Express. Create a new Class Library project (File - New Project, Class Library) named MyPane.

2. In the Solution Explorer window, right click Class1.vb and select Delete.

3. In the Solution Explorer window, right click MyPane and select Add - User Control. Set the control name to MyPaneControl and click Add.

4. In the MyPaneControl.vb design window, add a button (Toolbox, Common Controls, Button) and a label (Toolbox, Common Controls, Label). Double click the button to add a handler for the button's Click event.

5. In the Solution Explorer window, right click MyPane and select Add Reference. In the Add Reference dialog, switch to the Browse tab, locate the Manifold installation folder, select Manifold.Interop.dll and Manifold.Interop.Scripts.dll and click OK.

6. In the MyPaneControl.vb code window, select all text and replace it with:

```
Public Class MyPaneControl
 Implements Manifold.Interop.Scripts.IEventsConnection

 Dim app As Manifold.Interop.Application

  Private Sub Button1_Click( _
      ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles Button1.Click
      If Not (app Is Nothing) Then
          app.MessageBox("The document contains " + _
              app.ActiveDocument.ComponentSet.Count.ToString()
+ _
              " component(s).")
      End If
  End Sub

  Public Sub ConnectEvents( _
      ByVal ev As Manifold.Interop.Scripts.Events) _
      Implements
Manifold.Interop.Scripts.IEventsConnection.ConnectEvents
      AddHandler ev.DocumentClosed, AddressOf
Document_Changed
      AddHandler ev.DocumentCreated, AddressOf
Document_Changed
      AddHandler ev.DocumentOpened, AddressOf
Document_Changed
      AddHandler ev.DocumentSaved, AddressOf Document_Changed
  End Sub
```

```
  Private  Sub  Document_Changed(  _
      ByVal  sender  As  System.Object,  _
      ByVal  args  As
Manifold.Interop.Scripts.DocumentEventArgs)
      app  =  args.Document.Application
      Label1.Text  =  args.Document.Path
  End  Sub
End  Class
```

7. Invoke Project - Show All Files. In the Solution Explorer window, expand MyPane then MyProject and finally double click AssemblyInfo.vb. Locate the line which sets the value of the ComVisible attribute to False, and change it to set the value of the attribute to True:

```
<Assembly: ComVisible(True) >
```

8. Build the class library by using Build - Build MyPane. Make sure there are no build errors.

9. Save the code and the compiled binary by using File - Save All.

10. Launch Windows Explorer and locate the folder you saved the project to. Descend into bin\release, copy MyPane.dll and paste it into the Manifold configuration folder (usually, C:\Program Files\Manifold System\Config). If you do not have write permissions in the configuration folder, launch Manifold and change the file location for that folder to point to where you have write permissions.

11. In Windows Explorer, create a new XML file in the configuration folder and name it MyPane.xml for convenience (this file is different from MyPane.xml generated by Visual Basic). Drag and drop the created MyPane.xml into the opened instance of Visual Basic, set its text to:

```
<?xml version="1.0"?>
<xml>
  <addin>
    <name>My Pane</name>
    <command>
      <name>My Pane</name>
      <form>MyPane.dll;MyPane.MyPaneControl</form>
    </command>
  </addin>
</xml>
```

Save the MyPane.xml file. That's all!

To test the pane, launch a new instance of Manifold. Note that the Tools - Add-Ins menu contains a new command named My Pane. Invoke that command to show the pane. Create a new project. The pane label should become blank. Create a new drawing, then press the pane button.

The pane should report that the project .map file contains two components (a drawing and a table). Save the project. The pane label should update its text with the path to the .map file.

Add-ins and Forms

Add-ins can include forms, used as modeless panes. Forms can be either ActiveX controls or .NET controls.

To include a form into an add-in, create a command item using the <command> tag, supply the name of the form using the <name> tag and supply either the ProgID of the ActiveX control or the name of the .NET control using the <form> tag. The name of the .NET control should include the name of the assembly or path to the assembly module (either absolute or relative from the Config folder) and the name of the .NET class separated by a semicolon.

Tech Tips
Manifold cannot use any customizations if the .xml files do not contain XML that is exactly correct. A useful test before launching Manifold is to open any newly created or modified .xml file in Internet Explorer. Internet Explorer will show a correct .xml file in a simple text format. If Internet Explorer cannot parse the .xml file, Manifold won't be able to either.

Add-in forms implemented as .NET controls can subscribe to events fired by the Manifold core at runtime. To do this, include a reference to Manifold.Interop.dll and Manifold.Interop.Scripts.dll, implement the IEventsConnection interface from the latter dll and add handlers to the desired events in the ConnectEvents method of the interface.

Add-In Pane Example URL's

Copy all formats Tool
http://forum.manifold.net/Site/Thread.aspx?id=25027

Convert Track Points to Track Lines
http://forum.manifold.net/Site/Thread.aspx?id=29956

Column Name Add-in
http://forum.manifold.net/Site/Thread.aspx?id=33488

Manifold© COM Object Model Description and .NET Usage (theory)

http://69.17.46.171/Site/Thread.aspx?id=26683&ti=632927748277800000

adamw at 9/2/2006 6:20 AM (#26877)
The Manifold Object Model is a COM object model and .NET applications use it through an interop assembly named Manifold.Interop.dll. The interop assembly is generated from the COM

type library in Ext.dll by a standard tool in the .NET Framework SDK. The tool inspects the contents of the type library and generates .NET wrappers for the COM entities it finds.

COM entities do not map one to one to .NET entities, so sometimes the tool has to do something creative to make sure that the generated .NET module can do everything that would be possible to do via COM. When a COM entity maps to a .NET class, the tool makes sure the methods and properties generated from the type library do not shadow the methods and properties of the parent class. Thus, it renames any methods named ToString, since each .NET object inherits from System.Object and System.Object already has a method named ToString. It turns out that sometimes the tool renames a method or property even if there are no apparent naming conflicts, like in this particular case (the Document property). I do not know why exactly it does this, but one theory is that it does this in anticipation of future changes to the .NET Framework.

Manifold Object Model in VB 2005

http://forum.manifold.net/Site/Thread.aspx?id=31261&ti=633008396463230000

gxdata on 12/4/2006 8:07 AM (#31456)

Within Visual Studio:

- Once you have set a reference in VS2005 to Manifold.Interop, you have IntelliSense kicking in to help you, like other objects referenced within the VS IDE? That is very helpful.
- It is possible to use the Object Browser panel within Visual Studio, and have a look at the object model that way.

If you understand the rudiments of working within Visual Studio, it really is worthwhile trying some fairly simple things there, and then (if it is a better way to achieve what you want within the Manifold project (.MAP) environment, without an external application), to take the principles you have learned and do the same lessons or exercises with VBScripts.

Adding a Component to an Addin's Loaded ComponentSet

See the bolded text for main information:
http://forum.manifold.net/Site/Thread.aspx?id=29079

njengler on 10/19/2006 11:01 AM (#29080)

I am getting a strange error and I am wondering if it is
a) a Manifold bug, or
b) something strange in my code.
I am working on a VB.NET Add-In for Manifold. The following lines of code:

```
gComponentSet = gApplication.ActiveDocument.ComponentSet()
gDocument = gApplication.DocumentSet.Item(0)
mFolder = gDocument.NewFolder("Name")
```

gComponentSet.Add(mFolder)

Work perfectly if I open Manifold, load my Add-In (which opens a Pane) and then load a project (I have Manifold.Interop.Scripts.IEventsConnection implemented, so my Add-In knows when a project is opened).

However, if I open Manifold, load the project, and then load my Add-In I get the following error:

"Cannot add component to component set"

However, the folder is actually added, and Manifold seems to continue on fine after I close the error message box. I have searched out all the obvious things that I can think of that might cause this problem, and nothing seems to be different between the two scenarios described above.

Any help or thoughts would be greatly appreciated

willh on 10/20/2006 10:01 AM (#29150)

You don't add a component to the document's ComponentSet; it is part of that set by way of it being created through the Document.

njengler on 10/20/2006 8:38 PM (#29165)

willh ... you are suggesting that my code is redundant then? If so, then this line of code

mFolder = gDocument.NewFolder("Name")

negates the need for

gComponentSet.Add(mFolder)

That would be great news. However, the other issue, as described, how different outcomes occur depending on the sequence of events I take inside manifold (i.e. one order of events raises the error, the other does not) is not addresed?

Is there a reason behind this? Am I missing something?

Thanks!

vlacour on 10/21/2006 3:13 AM (#29168)

Seems to me that when you load the add-in, the original project might not be the active document anymore; you might want to:
first identify your document by
gDocument = gApplication.NewDocument("name_of_document", bReadOnly)
  or
gDocument.Open("name_of_document", bSaveChanges, bReadOnly)
  then
mFolder = gDocument.NewFolder("Name")

Hope it helps.

vincent

njengler on 10/26/2006 6:33 AM (#29346)
The problem was with the redundancy in my code: creating the folder object (which also adds it to the componentset) and then trying to add it to the componentset again. This was causing Manifold to raise the error (due to unique component name requirements I am guessing).

Thanks for your help!

Calling EventArgs without using the OnClick Method

http://forum.manifold.net/Site/Thread.aspx?id=22306

rbeemer on 6/7/2006 2:45 PM (#22307)
I am trying to call the EventArgs object in a script other than OnClick. When the script is run the message "Object required: EventArgs" appears. How do I access the EventArgs object?

```
If EventArgs.HasObject Then
  varX = EventArgs.LocationNative.X
  varY = EventArgs.LocationNative.Y
End If
```

adamw on 6/8/2006 6:02 AM (#22344)
You can not. What are you trying to do?

rbeemer on 6/8/2006 8:26 AM (#22370)
I need to have a user select a location in a map window, and also have a user select an object, or objects in a specific sequence, from within the script. Basically, I need the information that the EventArgs provides from a mouse click.

adamw on 6/9/2006 5:59 AM (#22440)
Why then are you not using the OnClick script?

rbeemer on 6/9/2006 10:50 AM (#22474)
Because this is part of a complex process that can't be running every time a mouse is clicked. Unless the OnClick can pass values or reference to another script?

dmbrubac on 6/10/2006 6:35 AM (#22506)
Here is some partial code that lets you get at the selected objects in a drawing (note that you are actually looking for a table). You can easily filter what you do and do not want to process. This is VB.NET code to clearly demonstrate a concept, not necessarily a best practices way of doing things.

```
Private Sub ConnectEvents(ByVal ev As Manifold.Interop.Scripts.Events) _
  Implements Manifold.Interop.Scripts.IEventsConnection.ConnectEvents
  AddHandler ev.ComponentSelectionChanged, AddressOf Component_SelectionChanged

End Sub


Private Sub Component_SelectionChanged( _
      ByVal sender As System.Object, _
      ByVal args As Manifold.Interop.Scripts.ComponentEventArgs)

  Label1.Text = "Selection Changed in " & args.Component.Name

  If TypeOf args.Component Is Manifold.Interop.Table Then
    ' cast the sucker to a table
    Dim tblOurTable As Manifold.Interop.Table
    tblOurTable = CType(args.Component, Manifold.Interop.Table)
    If tblOurTable.Selection.Count = 0 Then
      Label1.Text = Label1.Text & vbNewLine & " No Objects Selected"
    Else
      Dim i As Integer
      For i = 0 To tblOurTable.Selection.Count
        ' stringbuilder is obviously better here
        Label1.Text = Label1.Text & vbNewLine & " Selected ID: " & _
                      tblOurTable.Selection(i).ID.ToString
      Next
    End If
  End If

End Sub
```

VB.Net Samples

http://forum.manifold.net/Site/Thread.aspx?id=32243&ti=633017663024830000

diwakar_rs on 12/15/2006 3:41 AM (#32244)

I am very new to manifold migrated from asp maps(GIS engine).
Are there any sample applications on manifold in VB.Net (ASP.Net) or related to IMS.

KlausDE on 12/15/2006 7:55 AM (#32259)

Search this forum and the archive for "MapControl" OR "ComponentControl". The ComponentControl is the object you will use for an application. Be sure to notice the few threads dealing with events. Here you find a collection of examples using ComponentControl
http://69.17.46.171/Site/Thread.aspx?id=27477&ti=632938400526500000

ManifoldControl.zip
http://forum.manifold.net/Attachments/54/27483/ManifoldControl.zip

DemoMap.zip
http://forum.manifold.net/Attachments/58/27485/DemoMap.zip

MapViewer.zip
http://forum.manifold.net/Attachments/52/27527/ManifoldViewer.zip

adamw on 12/15/2006 7:58 AM (#32260)

Well, are you looking to create a web site or an application? If you are looking to create a web site, you would do good by examining the web site examples on the Free Stuff page. None of these examples is in VB .NET (which is unfortunate), but the VB .NET code would be pretty similar and if you are into web applications, you'd better learn JavaScript for all that AJAX stuff anyway. :-)

Loading and Compiling the MapViewer Project – Converting to VB2005

http://forum.manifold.net/Site/Thread.aspx?id=33473

trondwe on 1/16/2007 3:11 AM (#33520)

REFERENCE: http://69.17.46.171/Site/Thread.aspx?id=27477

i) First I upgraded the ManifoldViewer project from VS2003 to VS2005 using the inbuild wizard in Visual Studio.

ii) Deleted the old reference to Manifold and added reference to the 7x library

Still did not work - could not see the FormMain dialog in the designer, just obscure messages.

iii) Deleted the old reference to ActiveX Manifold control and added it again to the Toolbox - still problems

iv) Added a new form and added the 7x Manifold control to that form

v) Deleted the new form

Now I could see FormMain in the designer !!

Then I had to alter the sourcecode:

- Fully qualify "DialogResult.Ok"
- Alter method .ctlRefresh() to .Refresh() for the Manifold© control three places.

Now I could start the application - puh

With the ManifoldViewer application I could read a .map file and I could select polygons in that .map file

BUT I am not able to zoom in - is that intended behaviour of the application ??

I would really appreciate if you could look into this one also !

Regards, Trond

FOLLOWUP NOTE L. Ketch

There also seems to be a problem with code that should now reside in FormMain.Designer.vb being originally included in FromMain.vb. There are double instances of constants and subroutines that have to be deleted.

I deleted and moved some of the problem code. There is now a working VB.net 2005 version stored at:
C:\Documents and Settings\Lorne_Limited\My Documents\Manifold\ActiveX_Example\ManifoldViewer

Note that the DemoMap example has also been converted and resides at:
C:\Documents and Settings\Lorne_Limited\My Documents\Manifold\ActiveX_Example\DemoMap

7x COM/Registry problem?

http://forum.manifold.net/Site/Thread.aspx?id=31598

cwarner on 12/6/2006 3:12 PM (#31599)

(example dimensioning of various Manifold© objects)

```
Dim manApp As Manifold.Application
Dim manDoc As Manifold.Document
Dim manComps As Manifold.ComponentSet

manApp = CreateObject("Manifold.Application")
  ' Using  "= New Manifold.Application" above has same behaviour
manDoc = manApp.ActiveDocument
manComps = manDoc.ComponentSet

Dim proc As MyLibrary.Module1 = New MyLibrary.Module1
proc.Components = manComps
proc.DoStuff
```

This always hangs up on the CreateObject line, eventually throwing the exception "Cannot Create ActiveX Object." This exact same code used worked just fine with Manifold 6.0, 6.5, and I believe 7.0 (I may have not actually used the now-problematic functionality in the brief interim between installing 7.0 and 7x).

Progressbar Object

njengler on 7/2/2006 2:35 PM (#18931)

Does anyone know the code to successfully handle a user-enacted cancel of the ProgressBar object (in VB.NET, for an Add-in to Manifold 7)?

adamw on 7/18/2006 9:48 AM (#24511)

I just tried this code and it seems to halt OK (both messages are shown when you cancel):

```
'VB.NET
Imports Manifold.Interop.Scripts
Imports Manifold.Interop
Imports System.Threading

Class Script
  Shared Sub Main
    Dim progress As Progress = Context.Application.NewProgress
    progress.MaxPosition = 100
    progress.Text = "Lengthy operation"
    progress.Start()
    Do While progress.Position < progress.MaxPosition
      progress.Position = progress.Position + 1
      progress.Text = "Lengthy operation (" & progress.Position & "% done)"
      If Not progress.Update Then
        Context.Application.MessageBox("Canceled", "Script")
        Exit Do
      End If
      Thread.Sleep(100)
    Loop
    progress.Stop()
    Context.Application.MessageBox("After the loop", "Script")
  End Sub
End Class
```

Using the VBA Programming Environment

http://www.georeference.org/DNN31/KnowledgeBase/Articles/UsingVBA/tabid/87/Default.aspx?PageContentID=1

By Robert Heitzman

You will most commonly use Manifold scripting from within the Manifold program. However the Manifold .Map projects can be accessed from other programming environments that have a lot to offer both the beginning programmer and those that are pushing the Manifold object model to its limits.

This topic will show you how to use Word VBA to access a .Map file. Word itself will not be used, just the VBA IDE (Integrated Development Environment).

The same concepts apply to other Office VBA programs (Excel being one of the most useful, as well as Access) and to 'real' programming environments like Visual Basic and its big brother Visual Studio. Older version as well as the current .Net version can be used.

To prepare for this, note the full Windows path to a Manifold. On my PC the path is:

D:\GIS_DATA\Scripts\ScriptingTutorial.map

Note the location on your PC and use that path in the code on the next page.

Office VBA IDE

If you are using Word to view this document all you need to do to access the Office VBA IDE is key Alt-F11 (Visual Basic from the Tools, Macro menu can be used as well as the Visual Basic toolbar).

Off to the right of the IDE you will see a Projects window similar to the one used in Manifold. Double click the ThisDocument component in the Microsoft Word Objects folder. A blank screen will appear in the main window.

Enter this code:

```
Option Explicit
Sub Tutorial()
   Application.MessageBox "Hello World!", vbExclamation, "Hey!!!!"
End Sub
```

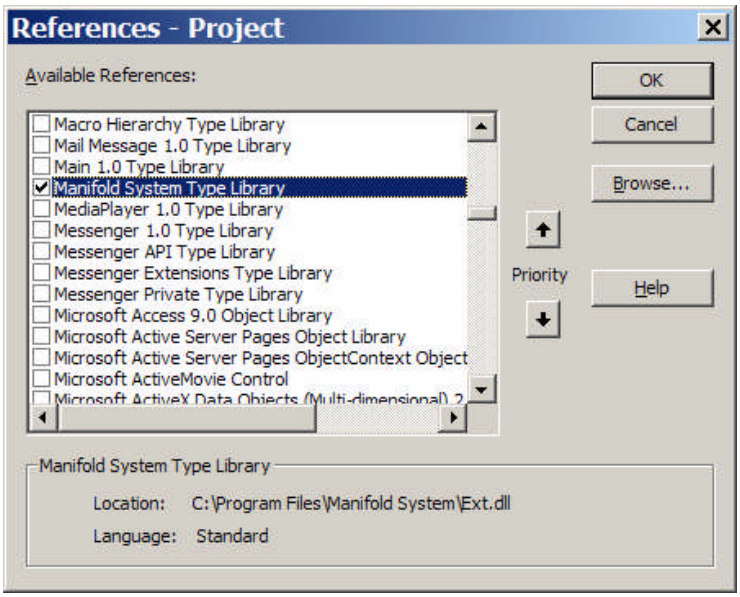
Place the mouse cursor inside the Sub Tutorial (on the Application.Messagebox statement) and hit F5 or click the run button on the toolbar.

As you were typing the code the IDE was making suggestions and filling in the blanks – very handy! This feature is called IntelliSense and I predict you will become addicted to this feature!

The Option Explicit *directive* forces you to declare variables before you use them in the code. This restriction requires you to understand what the variables will represent before they are used. This can help you over the Manifold learning curve quickly and the discipline imposed is good for the quality of the final product. You can program without Option Explicit but Excel will just use generic objects and that may get you into trouble. However the biggest benefit to declaring variables is that IntelliSense can work to its highest potential.

The Manifold System Type Library

In order to access Manifold project files the IDE needs to be provided a description of the objects it will encounter. To accomplish this use the Tools, References menu. Scroll down to the Manifold System Type Library in the list (click M to scroll down). Check the box and click OK.



Enter the following code:

```
  Sub Tutorial()
      Dim manApp As Manifold.Application
      Dim manDoc As Manifold.document
      Dim filePath As String

      Set manApp = Manifold.Application
      filePath =
"D:\GIS_DATA\Scripts\ScriptingTutorial.map"
' path to your file here
      Set manDoc =
manApp.DocumentSet.Open(filePath, True)
      Application.MessageBox "Path to .map
file:" & manDoc.Path, vbExclamation, "VBA
Rocks!"
  End Sub
```

Since we want to use Intellisense we have used the Option Explicit directive and declared the variables we will need, specifying the specific data type.

Note that here we declared a specific variable (Dim manApp As Manifold.Application) to hold a reference to the Manifold.Application object which was not required in the VBScript environment. The reason for that is if we just type Application what we will get in the IDE is Word.Application not Manifold.Application.

Dim filePath As String provides us a place to store text, in this case the path to the document we are working on. There are a wide variety of ways to find which file to open but here we will just hardcode the path to the file. Change the code above to point to the location of your training .map file. We could have entered the hardcoded path inside the Open method directly.

To test the code, this time place the mouse cursor anywhere in the Tutorial Sub and hit F8. This will single step us into the subroutine. Hit F8 again to go to the next statement. Place the mouse cursor over filePath in the highlighted statement. Hit F8 again and move the mouse over filePath again – neat huh!

You can hit F5 anytime to run at normal speed through the rest of the code. Note that Word's (or Excel's) main document window may not act as you expect while you are stepping through code. Hit the Reset button on the toolbar before returning to the main document.

Caution: Word (the VBA IDE) and Manifold (the program) should not have the same .map file open at the same time. Manifold project files do not support multi-user access. Neither program may complain when they both open the same file but what happens when the programs close the .map file is unpredictable.

Depending on what mistakes you make while writing code the IDE may become unstable and freeze up or fail to do anything but stop running. Of course you may be able to avoid this by never making mistakes, but as an alternative save your work early and often. Keep backup copies of your .map files as well.

Trapping Errors

Word will complain at times when you make typing errors but once Word thinks things are okay we are not out of the woods yet. We can make a blunder the IDE cannot detect. To help us deal with this we will put in some error handling. Add the marked lines to your code:

```
Option Explicit
Sub Tutorial()
    On Error GoTo Local_Err                         '<--- Add
    Dim manApp As Manifold.Application
    Dim manDoc As Manifold.document
    Dim filePath As String

    filePath = 1 / 0 ' cause an error                          '<--- Add
    Set manApp = Manifold.Application
    filePath = "D:\GIS_DATA\Scripts\ScriptingTutorial.map" ' path to your file here
    Set manDoc = manApp.DocumentSet.Open(filePath, True)
    Application.MessageBox "Path to .map file:" & manDoc.Path, vbExclamation, "VBA
Rocks!"
Local_Exit:                          '<--- Add
    Exit Sub                          '<--- Add
Local_Err:                          '<--- Add
    Application.MessageBox VBA.Err & " " &
VBA.Err.Description                             '<--- Add
    'Resume Local_Exit                            '<--- Add
    ' while testing comment out the statement above                          '<--- Add
    Stop ' F8 to return to statement with the error                          '<--- Add
    Resume                          '<--- Add
End Sub
```

Press F5 to test. You should see the error message and when you click OK you will hit the Stop statement. Press F8 to step back to the statement in error. Unlike the VBScript IDE you can edit the code while you are stepping through the code. Place a single quote in front of the statement in error and use F8 or F5 to continue. You can right click while stepping and move the highlight to a different statement.

It is always a good idea to cleanup before the final exit. Close all objects that have a Close method and any object that was set using the Set VBA command should be set to Nothing. Here is the code with the error statement removed.

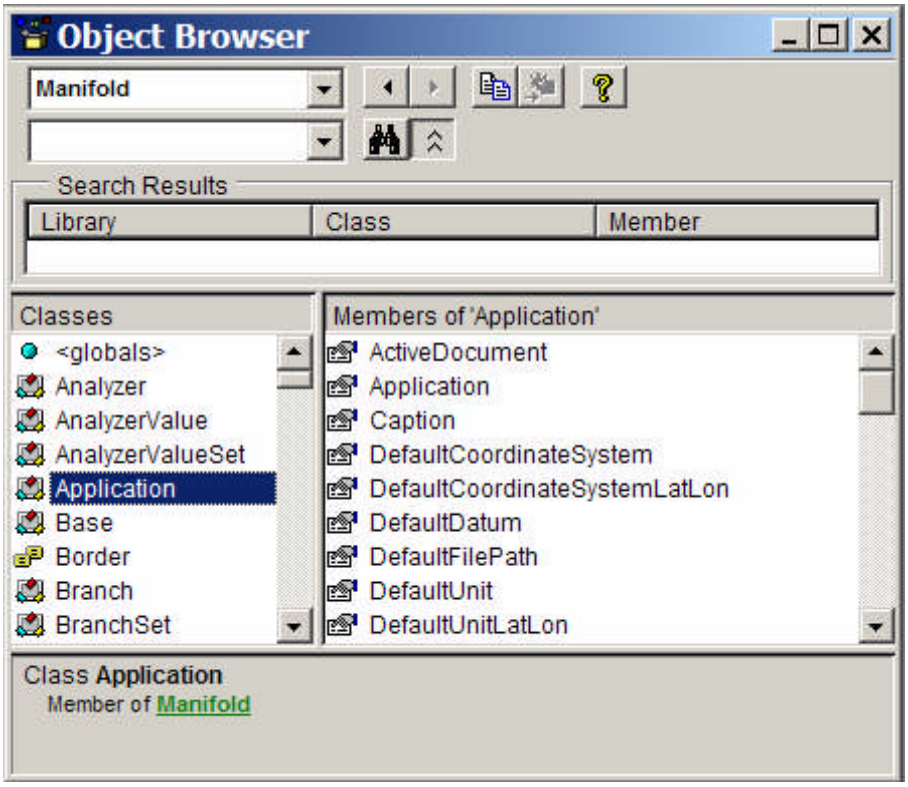
```
Option Explicit
Sub Tutorial()
    On Error GoTo Local_Err
    Dim manApp As Manifold.Application
    Dim manDoc As Manifold.document
    Dim filePath As String

    Set manApp = Manifold.Application
    filePath = "D:\GIS_DATA\Scripts\ScriptingTutorial.map" ' path to your file here
    Set manDoc = manApp.DocumentSet.Open(filePath, True)
    Application.MessageBox "Path to .map file:" & manDoc.Path, vbExclamation, "VBA
Rocks!"
Local_Exit:
    On Error Resume Next ' block error handling during cleanup to avoid
loop                          '<--- Add
    manDoc.Close                            '<--- Add
    Set manDoc = Nothing                          '<--- Add
    Set manApp = Nothing                          '<--- Add
    Exit Sub                          '<--- Add
Local_Err:
    Application.MessageBox VBA.Err & " " & VBA.Err.Description
    Resume Local_Exit                           '<--- Add
    ' while testing comment out the statement above
    Stop ' F8 to return to statement with the error
    Resume
End Sub
```

Additional IDE Tools – Object Browser, Immediate Window, Locals Window

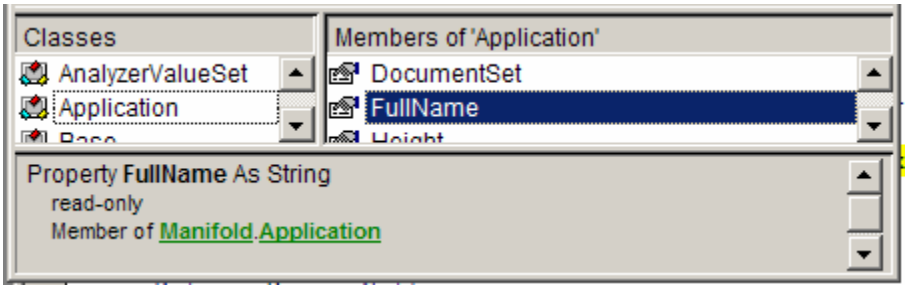
On the IDE toolbar click the Object Browser icon. 

You should see this, probably full screen, after you choose Manifold from the dropdown list in the top left:



Click on Application and scroll through the Members of 'Application' list.

Basically what you have here is a concise list of properties and methods for all of the Manifold objects. Note that some items in the Members pane are simple properties that return text or numbers and other properties are other objects. When you click on items in the Members pane the box at the bottom of the dialog will tell you what type of object is returned to the code that references the object.



For example scroll down the list until you come to Application.FullName. You will see it returns a String (text) and it is read-only. Click DocumetSet just above and you will note that it returns an object of type ObjectSet.

Immediate Window

Open you code window again and step into the code using F8. Step to just past the Set manApp =… statement. Then select View, Immediate Window from the menus. In the new window type ?manApp.FullName.

```
 ?manapp.FullName Manifold System 5.50 Enterprise Edition SP2
(build 348)
```

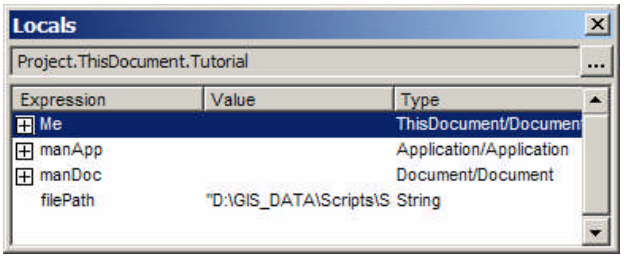
The ? will display what it can – text and numbers. It will not display objects. But you can keep drilling down the objects until you come to some text:

```
?manapp.DefaultCoordinateSystem.Name Orthographic
```

Note that for this to work you will have had to step into the code and gone past the point where manApp is set to Manifold.Application in the script used above.

Locals Window

Stop the execution of the code by hitting the Reset button on the toolbar. Click the menu View, Locals Window, place the mouse cursor in the code and press F8. The values will fill in as you step through the code. You can drill down to see the values of thousands of properties.



A window like this one is available in Manifold – the Variables Pane. I believe you must have the Debug Option to see the Variables Pane.

Done for now!

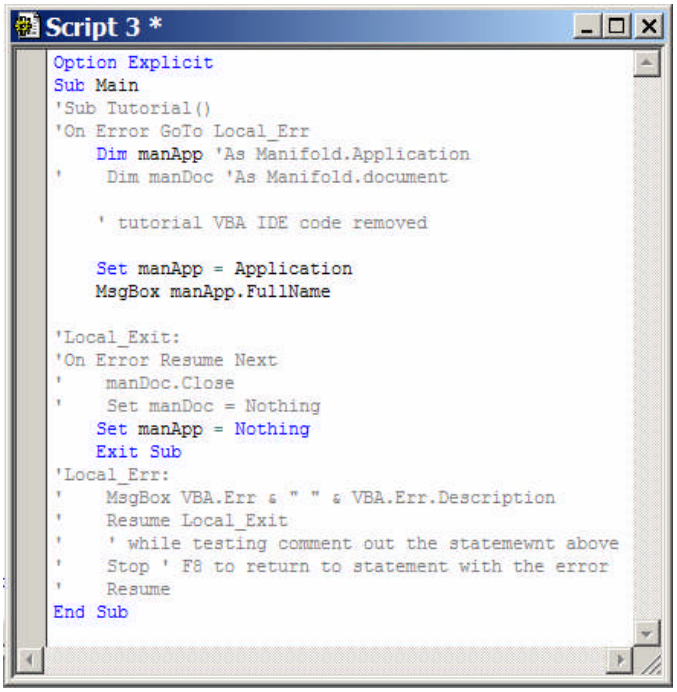
Save your work for the next article, Exploring the Manifold Object Model.

The code you develop can be copied and pasted into Manifold scripts with a few modifications. VBScript supports Option Explicit and the Dim statement but not the 'As' clause. Option Explicit is still recommended to avoid minor typos from creating hard to find errors.

For example if Option Explicit is off, code errors like this one can be hard to debug:

```
Dim manApp
Set manAppp = Application
Application.MessageBox
manApp.FullName
```

Note the two different spellings of



manApp. The Application object isn't the best example but you get the idea. If Option Explicit is on the second statement would generate an error.

Of course the point of the samples above is to show how to use the VBA IDE, so much of that code is not useful back in Manifold. If we had dug further into the object model almost all of that code would have been useful.

On the left the items commented out are not support by VBScript:

- Sub Main() required
- On Error Goto n/a
- Dim As not fully supported
- Line labels not supported
- Resume not supported

Create a MAP Project File and Drawing through COM or .NET

Manifold©-L

Author: Adam Wachowski

Date: 12-31-2005 07:23

Subject: RE: Native Manifold data exchange format

http://lists.directionsmag.com/discussion/read.php?f=29&i=41227&t=41227

> > > Does Manifold provide a native ASCII data exchange format?
> >
> > No. Why not exchange data using MAP files?
>
> To answer your question, I will use an example taken from the
> National Hurricane Centre's tropical cyclone tracks database ...
>
> The only way I have found to parse a complex file like this and create
> line segment storm tracks is to build a dedicated decode program and
> write the details to a MID/MIF file (I'm a 12 year Mapinfo user
> switching over to Manifold). I could then import the line segments
> and view and analyse the tracks. I don't know how one would write to
> the binary Manifold MAP file format.
> http://lists.directionsmag.com/discussion/read.php?f=29&i=41166&t=41166

You could produce MAP files using the Manifold object model, through COM or .NET.

The following C# code creates a blank MAP file, adds a drawing, adds some points, and saves the result:

```
using System;
using System.Reflection;
using Mfd = Manifold.Interop;

namespace Test
{
  class App
  {
    [STAThread]
    static void Main(String[] _args)
    {
      Mfd.Application _app = new Mfd.ApplicationClass();
      Mfd.Document _doc = _app.NewDocument("", false);
      Mfd.Drawing _drw = _doc.NewDrawing("Drawing", Missing.Value, false);
      _drw.ObjectSet.Add(
        _app.NewGeom(Mfd.GeomType.GeomPoint, _app.NewPoint(1, 1)));
      _drw.ObjectSet.Add(
        _app.NewGeom(Mfd.GeomType.GeomPoint, _app.NewPoint(2, 2)));
      _doc.SaveAs(@"C:\Documents\Test.map");
    }
  }
}
```

Adam Wachowski

Creating a View with a Specific Scale

http://69.17.46.171/Site/Thread.aspx?id=20558&ti=632812323737970000

paquet at 4/19/2006 5:35 AM (#20559)

Hi,

I'm accessing a .map file using VB.Net. I'd like to create a View for a component, at a desired cartographic scale (e.g. 1: 10,000). When we create a view it requires a scale, but it's the scaleInternal. It seems there is no explanation in Manifold Manual on scaleInternal.

How can we create a view with desired cartographic scale?

paquet

p.s. I know how to do this using Component windows, but since the .map file is accessed in the background with VB.Net, windows can't be used.

paquet at 4/21/2006 4:06 PM (#20596)

I found the answer to this myself. Here's how (VB.Net) (If it's useful for someone):

```
' ************************************************************

Dim surfA as surface = comps("Surface A")
' -- Calculate internal scale
Dim dScale As Double = 50000 ' 1:50,000
Dim dScaleInternal As Double = scale2scaleInternal(dScale, surfA)
' -- Create the view
Dim ViewSet As ViewSet = surfA.ViewSet
Dim pt As Point = surfA.PixelSet.Box.Center
Dim View As View = _
   app.NewView("View @" + dScale.ToString, dScaleInternal, pt.X, pt.Y)

ViewSet.Add(View)

doc.save()

Public Function getDisplayResolution() As Double
   Dim frmTest As Windows.Forms.Form = New Windows.Forms.Form
   frmTest.AutoScaleMode = Windows.Forms.AutoScaleMode.Dpi
   getDisplayResolution = frmTest.CurrentAutoScaleDimensions.Width
   frmTest = Nothing
End Function

' ************************************************************

Public Function scale2scaleInternal(ByVal dScale As Double, ByVal cmp As Component) As
Double
'***Currently works for drawings and surfaces in meters
   Dim dDPI As Double = getDisplayResolution()
   Dim dInchInMeter As Double = 0.0254 'One inch is 0.0254 meter
   Dim dUnitsPerDot As Double = _
      dInchInMeter / dDPI ' 1:1 scale gives a scaleInternal of dUnitsPerPixel

   ' -- Find what is the number of pixels/unit for that surface
   Dim rectBB As Rect

   If cmp.Type = ComponentType.ComponentSurface Then
      Dim surf As Surface = cmp
      rectBB = surf.PixelSet.Box
   Else
      rectBB = cmp.objectset.box
   End If

   Dim cs As CoordinateSystem = cmp.CoordinateSystem
   Dim dLocalOffsetX As Double = _
      cs.ParameterSet.Item(cs.ParameterSet.ItemByName("localOffsetX")).Value
   Dim dLocalScaleX As Double = _
      cs.ParameterSet.Item(cs.ParameterSet.ItemByName("localScaleX")).Value
   'RWU: Real World Units (m,feet,...)
   Dim dXMaxRWU As Double = dLocalOffsetX + (rectBB.XMax * dLocalScaleX)
   Dim dWidthRWU As Double = dXMaxRWU - (dLocalOffsetX + _
         rectBB.XMin * dLocalScaleX) 'Width in Real World Units

   ' -- Use offsets and scales of coordinate system to calculate pixel size.
   Dim dUnitPerPix As Double = _
       dWidthRWU / rectBB.Width ' Pixel size (width) of the surface
                                ' in the used unit (meter, ...)
```

```
   scale2scaleInternal = dScale * dUnitsPerDot / dUnitPerPix
End Function

' ********************************************************
```

Cancelling the ProgressBar Object from VB.NET

http://69.17.46.171/Site/Thread.aspx?id=18930&ti=632888409132730000

njengler at 7/2/2006 2:35 PM (#18931)

Does anyone know the code to successfully handle a user-enacted cancel of the ProgressBar object (in VB.NET, for an Add-in to Manifold 7)? I include the following line of code in my loop, however it does not seem to be correct, as the ProgressBar does not halt and disappear (rather it throws an error on subsequent lines past where I expected it to have exited the loop):

```
If mProgress.Update = False Then
  Exit For
End If
```

Thanks for the help
N

adamw at 7/18/2006 9:48 AM (#24511)

I just tried this code and it seems to halt OK (both messages are shown when you cancel):

```
'VB.NET
Imports Manifold.Interop.Scripts
Imports Manifold.Interop
Imports System.Threading

Class Script
  Shared Sub Main
    Dim progress As Progress = Context.Application.NewProgress
    progress.MaxPosition = 100
    progress.Text = "Lengthy operation"
    progress.Start()
    Do While progress.Position < progress.MaxPosition
      progress.Position = progress.Position + 1
      progress.Text = "Lengthy operation (" & progress.Position & "% done)"
      If Not progress.Update Then
        Context.Application.MessageBox("Canceled", "Script")
        Exit Do
      End If
      Thread.Sleep(100)
    Loop
    progress.Stop()
    Context.Application.MessageBox("After the loop", "Script")
  End Sub
End Class
```

Programming an Add-In Toolbar DLL in VB.NET

http://69.17.46.171/Site/Thread.aspx?id=26739&ti=632927749818570000

adamw at 9/2/2006 6:20 AM (#26882)

You can create an add-in pane with a menu and have menu items launching other forms, modal or modeless.

You **cannot** create a visual tool like, say, Select Box, or intercept the output of the existing visual tool. You **can**, however, create your own application with an instance of the Manifold ActiveX control and create your own visual tools that would operate in that control.

How to Create a Component View at the Desired Scale via VB.Net

http://69.17.46.171/Site/Thread.aspx?id=20558

paquet on 4/19/2006 5:35 AM (#20559)

I'm accessing a .map file using VB.Net. I'd like to create a view for a component, at a desired cartographic scale (e.g. 1:10,000). When we create a view it requires a scale, but it's the ***scaleInternal***. It seems there no explanation in Manifold Manual on scaleInternal.

How can we create a view with desired cartographic scale?

paquet

p.s. I know how to do this using component windows but since the .map file is accessed in the background with VB.Net, windows can't be used.

paquet on 4/21/2006 4:06 PM (#20596)

I found the answer to this myself. Here's how (VB.Net) (If it's useful for someone):

```
'*****************************************

dim surfA as surface = comps("Surface A")

' -- Calculate internal scale
Dim dScale As Double = 50000 ' 1:50,000
Dim dScaleInternal As Double = scale2scaleInternal(dScale, surfA)

' -- Create the view
Dim ViewSet As ViewSet = surfA.ViewSet
Dim pt As Point = surfA.PixelSet.Box.Center
Dim View As View = app.NewView("View @" + dScale.ToString, dScaleInternal, pt.X, pt.Y)

ViewSet.Add(View)
```

```
doc.save()

'*****************************************

Public Function getDisplayResolution() As Double
  Dim frmTest As Windows.Forms.Form = New Windows.Forms.Form
  frmTest.AutoScaleMode = Windows.Forms.AutoScaleMode.Dpi
  getDisplayResolution = frmTest.CurrentAutoScaleDimensions.Width
  frmTest = Nothing
End Function

'*****************************************

Public Function scale2scaleInternal( _
       ByVal dScale As Double, _
       ByVal cmp As Component) _
          As Double

     '***Currently works for drawings and surfaces in meters
  Dim dDPI As Double = getDisplayResolution()
  Dim dInchInMeter As Double = 0.0254 'One inch is 0.0254 meter

     ' 1:1 scale gives a scaleInternal of dUnitsPerPixel
  Dim dUnitsPerDot As Double = dInchInMeter / dDPI

     ' -- Find what is the number of pixels/unit for that surface
  Dim rectBB As Rect
  If cmp.Type = ComponentType.ComponentSurface Then
     Dim surf As Surface = cmp
     rectBB = surf.PixelSet.Box
  Else
     rectBB = cmp.objectset.box
  End If

  Dim cs As CoordinateSystem = cmp.CoordinateSystem
  Dim dLocalOffsetX As Double = _

  cs.ParameterSet.Item(cs.ParameterSet.ItemByName("localOffsetX")).Value

  Dim dLocalScaleX As Double = _
     cs.ParameterSet.Item(cs.ParameterSet.ItemByName("localScaleX")).Value

     'RWU: Real World Units (m,feet,...)
  Dim dXMaxRWU As Double = dLocalOffsetX + (rectBB.XMax * dLocalScaleX)

     'Width in Real World Units
  Dim dWidthRWU As Double = dXMaxRWU - (dLocalOffsetX + rectBB.XMin * dLocalScaleX)

     ' -- Use offsets and scales of coordinate system to calculate pixel size.

     ' Pixel size (width) of the surface in the used unit (meter, ...)
  Dim dUnitPerPix As Double = dWidthRWU / rectBB.Width
  scale2scaleInternal = dScale * dUnitsPerDot / dUnitPerPix
End Function

'*****************************************
```

Build a Form via VB.net Code

http://forum.manifold.net/Site/Thread.aspx?id=20914&

adamw on 5/10/2006 10:28 PM (#20971)

There are several ways of creating custom forms in Manifold:

1. You can use form components. This way, you create a form component and insert the desired event handlers such as Button_Click into the attached script component. This is only available for non-.NET scripts, that is, VBScript and JScript.

2. You can use COM objects. This way, you create a COM object using the language and technology of your choice (VC++ / ATL, VC++ / MFC, VC++ / raw Win32 API, VB 6, VB .NET / WindowsForms, C# / WindowsForms, anything else), and call that COM object from within a script. This is available for both non-.NET and .NET scripts.

3. You can use various system and third-party libraries to build a form on the fly using code. For example, you can use WindowsForms to build forms in .NET scripts.

Example:

```
'VB .NET
Imports Manifold.Interop.Scripts
Imports System
Imports System.Drawing
Imports System.Windows.Forms

Class Script

  Shared Sub Main
    Dim f As New MyForm
    f.ShowDialog
  End Sub

  Class MyForm
    Inherits Form

    Public Sub New
      Dim b As New Button
      b.Location = New Point(20, 20)
      b.Text = "Say Hello"
      AddHandler b.Click, AddressOf B_Click
      Controls.Add(b)
      Text = "My Form"
    End Sub

    Private Sub B_Click(ByVal _sender As Object, ByVal _args As EventArgs)
      MessageBox.Show("Hi")
    End Sub
```

```
  End Class

End Class
```

Coordinate System Manipulation

Setting Current Projection of Images

http://69.17.46.171/Site/Thread.aspx?id=1213&ti=632114255400000000

jkelly on 2/3/2004 5:19 PM (#1214)

If you are importing images that have specific scale and offset values, then the usual "set current projection" scripts about don't take these values into account, resetting the origin as 0,0. The script below allows a bulk re-projection of images that are in this category.

```
{code - temp tag to be replaced}
'' AssignAMGCoordSysToImage
''
'' Assign a Current Coordinate System to images that have scale
'' and offset values.  This could be incorporated into your
'' Manifold© configuration as an Add-In menu or toolbar button.
'' See the "Add-Ins" topic

'' Uses all the parameters of the coordinate system
'' (in this case Australia AMG66 (55)) except the local scale and
'' offset values, which are obtained from the image before the
'' current projection is set.

' Author: James Kelly Date: February 4 2004 Version: 1.0.0
' jkelly@lesterfranks.com.au
' Platform: Manifold 5.5 SP2. Not tested on any other.
'
' Language: VBScript (Manifold System)
'
' Permission to use, modify and distribute this code is hereby granted,
' providing this entire comment section is included. No warranty is
' expressed or implied.

' ********************************************************************

Sub Main
  Dim Components, theComponent
  Dim I

  Set Components = Application.ActiveDocument.ComponentSet

  For I = 0 to Components.Count-1
    Set theComponent = Components.Item(I)
    If theComponent.TypeName = "Image" then
       AssignAMGCoordSysToImage(theComponent)
    End If
  Next

End Sub

' ********************************************************************

Function AssignAMGCoordSysToImage(theImage)
```

```
  Dim theCurrentCoordSys
  Dim CurrentParams
  Dim theAMGCoordSys
  Dim toSetParams

  Set theCurrentCoordSys = theImage.CoordinateSystem
  Set CurrentParams = theCurrentCoordSys.Parameters
  Set theAMGCoordSys = Application.newCoordinateSystem("Australia AMG66 (55)")
  Set toSetParams = theAMGcoordSys.Parameters
    ' Set the local scale and offset values of the new
    ' coordinate system to the existing values prior to
    ' setting the current projection.
  toSetParams(toSetParams.ItemByName("localScaleX")).Value = _
     currentParams(currentParams.ItemByName("localScaleX")).Value
  toSetParams(toSetParams.ItemByName("localScaleY")).Value = _
     currentParams(currentParams.ItemByName("localScaleY")).Value
  toSetParams(toSetParams.ItemByName("localOffsetX")).Value = _
     currentParams(currentParams.ItemByName("localOffsetX")).Value
  toSetParams(toSetParams.ItemByName("localOffsetY")).Value = _
     currentParams(currentParams.ItemByName("localOffsetY")).Value

    ' Set the current projection
  theImage.coordinateSystem = theAMGCoordSys

End Function

' ********************************************************************
```

Apply a New Projection to an Existing Drawing

Date: Thu 02/16/2006 2:47 AM
From: Adam Wachowski [adamw@manifold.net]
To: Manifold List
Subject: Re: [Manifold-l] Script: change current projection
http://lists.directionsmag.com/discussion/read.php?f=29&i=41556&t=41511

> I'd like to simply update the current projection params, that can be
> done via script and is extremely quick. ... Can you guide me here?

Create the projection you want and assign it to the desired component:

```
Sub Main
 Set csys = Application.NewCoordinateSystem("Mercator")
 Set drw = Application.ActiveDocument.ComponentSet("Drawing")
 drw.CoordinateSystem = csys
End Sub
```

Alternatively, reuse the projection of an existing component:

```
Sub Main
  Set drw1 = Application.ActiveDocument.ComponentSet("Drawing 1")
```

```
  Set drw2 = Application.ActiveDocument.ComponentSet("Drawing 2")
  drw2.CoordinateSystem = drw1.CoordinateSystem
End Sub
```

> What would really help neophytes like myself would be a "Learn Mode"
> where a series of GUI procedures would be stored into a script/macro
> as they were done. This could then be edited as needed and rerun.
> This is also a powerful learning tool as one can see the command
> sequence that was actually used and can build on/reuse this
> information. Just dreaming, my old DOS database system had this and
> it was very useful.

As it has been mentioned by others, this is already in the wishlist. If you want to stress the importance of having this feature in the system, email sales@manifold.net.

Adam Wachowski

Change Projections to Lat-Long (no offsets)

http://69.17.46.171/Site/Thread.aspx?id=24451&ti=632890871527770000

antoniocarlos at 7/19/2006 8:01 AM (#24556)

…..is there a script that takes all the drawings in a project and changes their projections to lat-long with no offsets of any kind so that this problem does not occur?

Why does Manifold use offsets rather than keep the coordinates as intrinsics?

Regards

adamw at 7/19/2006 9:32 AM (#24577)

Here you go:

```
'VBScript
Sub Main
  For Each c In Document.ComponentSet
    If c.Type = ComponentDrawing Then
      c.ProjectTo Application.DefaultCoordinateSystemLatLon
    End If
  Next
End Sub
```

All programs use offsets and scales, it is just that Manifold exposes them as part of the coordinate system. Other programs keep offsets and scales separate from the coordinate system (a world file vs a PRJ file).

XL VBA – Create a Components Projection Report

http://69.17.46.171/Site/Thread.aspx?id=166&ti=632079515400000000

rheitzman on 12/25/2003 12:19 PM (#747)

```
' Code developed under Excel VBA that evaluates the projected components
' (not all types as of yet) and places a "report" in the Immediate Window.

Option Explicit

' Tools, Reference, Manifold Type Library required

Sub ProjectionReport()

  Dim manApp As Manifold.Application
  Dim manDoc As Manifold.document
  Dim comp As Manifold.Component
  Dim components As Manifold.ComponentSet
  Dim report As Manifold.Comments
  Dim index As Integer
  Dim system As Manifold.CoordinateSystem
  Dim box As Manifold.rect
  Dim printBox As Boolean
  Dim objects As Manifold.ObjectSet
  Dim params As Manifold.CoordinateSystemParameterSet
  Dim s As String
  Dim I As Integer

  Set manApp = New Manifold.Application

  ' * supply your document path or use Appplication.GetOpenFileNAme from Excel

  Set manDoc = manApp.DocumentSet.Open("D:\GIS_Data\Scripts\ScriptingTutorial.map",
True)

  Set components = manDoc.ComponentSet

  Debug.Print manDoc.Name '& vbCrLf
  Debug.Print manDoc.Path '& vbCrLf
  Debug.Print Now() '& vbCrLf
  Debug.Print "------------------" '& vbCrLf

  ' -- traverse project components

  For index = 0 To components.Count - 1

    Set comp = components(index)

    Select Case comp.TypeName

       Case "Drawing", "Surface", "Map", "Image" ' ignore others

         Debug.Print vbCrLf & comp.TypeName & ":" & comp.Name '& vbCrLf
         Debug.Print " Note:" & comp.Note '& vbCrLf
```

```
        Debug.Print " Description:" & comp.Description '& vbCrLf

        Select Case comp.TypeName
          Case "Map"
          Case "Drawing"
            Debug.Print " Zoom:" & comp.ZoomMin & "/" & comp.ZoomMax '& vbCrLf
            Debug.Print " Precision:" & comp.Epsilon '& vbCrLf
          Case Else
            Debug.Print " Zoom:" & comp.ZoomMin & "/" & comp.ZoomMax '& vbCrLf
        End Select

        Set system = comp.CoordinateSystem

        Debug.Print " Coordinate System: " & system.Preset '& vbCrLf

        Debug.Print " Datum: " & system.Datum.Name '& vbCrLf

        Set params = comp.CoordinateSystem.Parameters

        For I = 0 To params.Count - 1
          With params(I)

            Select Case .Name ' supress select params, print others

              Case "majorAxis", "eccentricity", "centerX", "centerY", "centerZ"
                 ' ignore these 'cause I don't have a clue as to what they do

              Case "scaleX", "localScaleX", "scaleY", "localScaleY", "falseEasting",
"localOffsetX", "falseNorthing", "localOffsetY"
                 ' these are handled with direct calls latter

              Case Else ' unaccounted for params

                Debug.Print " +" & .Name & ": " & vbTab & .Value '& vbCrLf

            End Select

          End With
        Next

        With comp.CoordinateSystem.Parameters

           Debug.Print " Local Offset X/Y: " & vbTab & vbTab &
comp.CoordinateSystem.Parameters(.ItemByName("localOffsetX")).Value; '& vbcrlf
           Debug.Print ", " &
comp.CoordinateSystem.Parameters(.ItemByName("localOffsetY")).Value '& vbCrLf
           Debug.Print " Local Scale X/Y: " & vbTab & vbTab & vbTab &
comp.CoordinateSystem.Parameters(.ItemByName("localScaleX")).Value; '& vbcrlf
           Debug.Print ", " &
comp.CoordinateSystem.Parameters(.ItemByName("localScaleY")).Value '& vbCrLf
           Debug.Print " Scale X/Y(Ix): " & vbTab & vbTab & vbTab &
comp.CoordinateSystem.Parameters(.ItemByName("scaleX")).Value; '& vbcrlf
           Debug.Print ", " &
comp.CoordinateSystem.Parameters(.ItemByName("scaleY")).Value '& vbCrLf
           Debug.Print " False Easting/Northing: " & vbTab &
comp.CoordinateSystem.Parameters(.ItemByName("falseEasting")).Value; '& vbcrlf
```

```
            Debug.Print ", " &
comp.CoordinateSystem.Parameters(.ItemByName("falseNorthing")).Value '& vbCrLf
            Debug.Print " Lat/Lon(I)=" & system.Unit.LatLon '& vbCrLf
            Debug.Print " Units: " & system.Unit.Name & ", " & system.Unit.NickName &
", " & system.Unit.NickNameArea '& vbCrLf

            If system.Unit.LatLon = False Then
               Debug.Print " Scale to meters = " & system.Unit.Scale '& vbCrLf
            End If

         End With

         If comp.TypeName = "Drawing" Then

         End If

         Select Case comp.TypeName
           Case "Map"

           Case "Image", "Surface"
             Set box = comp.Pixelset.box
             printBox = True

           Case "Drawing"
             Set objects = comp.ObjectSet
             Debug.Print " Number of objects: " & CStr(objects.Count) '& vbCrLf

             If objects.Count Then
               Set box = comp.ObjectSet.GeomSet.box
               printBox = True
             End If

           Case Else
         End Select

         If printBox Then
           Debug.Print " Box: " '& vbCrLf
           Debug.Print " Width x Height: " & box.Width & " x " & box.Height '& vbCrLf
           Debug.Print " Area: " & box.Area '& vbCrLf
           Debug.Print " XMin(WestBC): " & box.XMin '& vbCrLf
           Debug.Print " XMax(EastBC): " & box.XMax '& vbCrLf
           Debug.Print " YMax(NorthBC): " & box.YMax '& vbCrLf
           Debug.Print " YMin(SouthBC): " & box.YMin '& vbCrLf
           printBox = False
         End If

    End Select

  Next

  ' report.Open

End Sub
```

Reset Orthographic Projection Centre to Map Window Centre

http://69.17.46.171/Site/Thread.aspx?id=2327&ti=632176735200000000

mdsumner on 4/15/2004 1:10 AM (#2328)

Hi, here's a simple script that some might be interested in. With a map in Orthographic projection open as the active window, this script will set the centre latitude and longitude of the map's /View/Projection/ to the centre of the window. If the active window is not a map nothing happens.

I'm using this to view an image of the world with the map in Orthographic projection - it gives a sort-of functionality to display a globe and rotate the view to where you want it without using the projection dialog - I use the "Center Point" tool on the open map window, then run the script to reset the projection's centre.

1. Create a map and set its /View/Projection/ to Orthographic
2. Use "Center Point" tool to recentre the map.
3. Click Run Script - "!" - with the script component highlighted in the Project Pane.
4. Repeat from 2 ;)

```
Option Explicit

Sub Main
  Dim Win, Map, vParams, Pt

  Set Win = Application.WindowSet.ActiveWindow
  If Win.Component.Type <> ComponentMap Then Exit Sub
  Set Map = Win.Component
  Set vParams = Map.CoordinateSystem.Parameters
  Set Pt = Win.LocationLatLon
  vParams.Item(vParams.ItemByName("CenterLat")).Value = Pt.Y
  vParams.Item(vParams.ItemByName("CenterLon")).Value = Pt.X
End Sub
```

Reassign Component Coordinate System & Datum - keep Original Scales & Offsets

```
' ********************************************************************

' Author: Michael D. Sumner Date: December 10 2003 Version: 1.0.1
' xxxx
' Platform: Manifold 5.5 SP2. Not tested on any other.
'
' Language: VBScript (Manifold System)
'
' Permission to use, modify and distribute this code is hereby granted, providing this
```

```
' entire comment section is included. No warranty is expressed or implied.

' ********************************************************************

' mdSumner provided some extra input to Bcowper on Sep 7, 2006 to convert the
' code to process Images.
' See GeoReference Thread:
' http://69.17.46.171/Site/Thread.aspx?id=27234&ti=632932921090100000

' The code below is a modified version of Michael Sumner's original.
' It has been reworked as a subroutine. L. Ketch Sep 2006


' ********************************************************************

Option Explicit


Sub Main
  Assign_CoordSys "Universal Transverse Mercator - Zone 17 (N)", _
                  "North American 1983 (Canada)" _
                   ComponentDrawing
End Sub

' ********************************************************************

Private Sub Assign_CoordSys ( _
         ByRef NewProjection, _
         ByRef NewDatum, _
         ByRef Comp_Type)

     ' NewProjection
     '    Projection String
     '    Ex. "Universal Transverse Mercator - Zone 17 (N)"

     ' New Datum
     '    Datum String
     '    Ex. "North American 1983 (Canada)"
     '
     ' Comp_Type
     '    One of the standard enumerations.
     '    Ex. ComponentDrawing, ComponentDrawing, ComponentSurface

     ' Use to reassign the Coordinate System and Datum of a
     ' Component when:
     ' 1. The imported component was not assigned the correct
     '    Coordinate System and Datum, But
     ' 2. The imported component Coordinate System parametres
     '    are OK (X and Y scales and Offsets).

  Dim Comp, Comps
  Dim Comp_CoodSys, Comp_CoodSys_Params
  Dim LocalScaleX, LocalScaleY, LocalOffsetX, LocalOffsetY

    ' Iterate over all components, setting the Current projection of
    ' each Image. Save the Offset and Scale parameters which are good.
```

```
  Set Comps = Document.ComponentSet

  For Each Comp in Comps

    If Comp.Type = Comp_Type Then

        ' Retrieve the CS and Parameters of the Image

      Set Comp_CoodSys        = Comp.CoordinateSystem
      Set Comp_CoodSys_Params = Comp_CoodSys.Parameters

        ' Save the Offset and Scale from the imported projection.

      LocalScaleX  = Comp_CoodSys_Params.Item("LocalScaleX").Value
      LocalScaleY  = Comp_CoodSys_Params.Item("LocalScaleY").Value
      LocalOffsetX = Comp_CoodSys_Params.Item("LocalOffsetX").Value
      LocalOffsetY = Comp_CoodSys_Params.Item("LocalOffsetY").Value

        ' Change the Current Projection of the Image to the desired
        ' one and respecify the original Scale and Offset.

      Comp_CoodSys.Load(NewProjection)
      Comp_CoodSys.Datum.Load(NewDatum)

      Comp_CoodSys_Params.Item("LocalScaleX").Value  = LocalScaleX
      Comp_CoodSys_Params.Item("LocalScaleY").Value  = LocalScaleY
      Comp_CoodSys_Params.Item("LocalOffsetX").Value = LocalOffsetX
      Comp_CoodSys_Params.Item("LocalOffsetY").Value = LocalOffsetY


    End If

  Next

End Sub

' ********************************************************************
```

Applying a Lat/Long Coordinate System to a Point

http://69.17.46.171/Site/Thread.aspx?id=17942&ti=632731991514300000

NOTE: the code below has an error as the author states. See fix at the bottom of the code (LAK)

nujseyer at 1/16/2006 4:06 PM (#17943)

I'm plotting a point from a table, I can't seem to apply the coordinate system.
Please help me...

Here is my code:

```
Sub Main
```

```
Set document = Application.ActiveDocument
Set components = document.ComponentSet

'Remove and Replace [Qry_Accomodation]
'------------------
i = 0
Do
if cstr(components(i).name) = "PlotXY" Then
components.Remove(i)
Else
i = i + 1
End If
Loop Until i = components.Count
'------------------

' -- create target drawing
Set drawing = document.NewDrawing("PlotXY", Application.DefaultCoordinateSystemLatLon)

Set objects = drawing.ObjectSet
Set table = drawing.OwnedTable
Set columns = table.ColumnSet
Set records = table.RecordSet

' -- obtain source data readers (My Position Table)
Set data = components(components.ItemByName("MyPosition"))
Set dataColumns = data.ColumnSet
Set dataRecords = data.RecordSet

' -- walk through data columns creating similar columns in target table
For columnIndex = 0 To dataColumns.Count-1
Set dataColumn = dataColumns(columnIndex)

Set column = columns.NewColumn()
column.Name = dataColumn.Name
column.Type = dataColumn.Type
column.Size = dataColumn.Size
columns.Add(column)
Next

' -- walk through data records creating random points in target drawing
For recordIndex = 0 To dataRecords.Count-1

' -- create lat/lon location
Set point = Application.NewPoint

' -- create drawing object
objects.Add Application.NewGeom(GeomPoint, point)

Set dataRecord = dataRecords(recordIndex)

' -- transfer fields
Set record = records(records.Count-1)
For columnIndex = 0 To dataColumns.Count-1
Set dataColumn = dataColumns(columnIndex)

record.Data(dataColumn.Name) = dataRecord.Data(dataColumn.Name)
Next
```

```
Next

'Add Component to Map
'--------------------
Set Dwg = Application.ActiveDocument
Set comps = Dwg.ComponentSet
Set map = comps("Map of Angeles")
Set mapLayerSet = map.LayerSet

Set gcps = comps("PlotXY")
Set newLayer = Dwg.NewLayer(gcps, 1)
mapLayerSet.Add newLayer

'Display Map
map.Open

End Sub
```

nujseyer at 1/18/2006 4:39 PM (#18011)

I've solve my problem. I just needed to insert some source code after:

```
*----------
' -- create lat/lon location
Set point = Application.NewPoint
*------------
point.X = my longitude
point.Y = my latitude
```

anyway, thanks to all....

Improving Performance when using CoordinateConverter.Convert()

http://69.17.46.171/Site/Thread.aspx?id=17995&ti=632736830017630000

nwdanner at 1/18/2006 1:02 PM (#17996)

Hey guys,

I'm new to the "new" forum and need some help. I' ve written a script which changes a drawing's coordinate system using the CoordinateConverter object. My VBA script (see below) works, but it is *magnitudes* slower than performing the same task manually using the GUI. For example, I have a fairly high point-detail drawing containing approximately 2200 objects. When I reproject the original drawing manually it takes roughly 7 seconds to finish. When I perform the same exact operation programmatically using the following script, it takes about 70 minutes!

Would someone look at the script below and let me know if I'm doing something wrong or can otherwise provide insight on how to achieve a higher level of performance.

Note, in my test the drawing being convertered is in lat/long and I'm converting it to Transverse Mercator, NAD 1983 using the script.

Thanks in advance for any help you guys can offer!

```
Option Explicit

Sub Main
Dim doc, windows, window, dwg
set doc = Application.ActiveDocument

Set windows = Application.WindowSet

' ensure there is at least one opened window
If windows.Count < 1 Then
Exit Sub
End If

Set window = windows.ActiveWindow

' ensure active window contains drawing
If window.ActiveComponent.Type <> ComponentDrawing Then
Exit Sub
End If

set dwg = window.ActiveComponent

' Ensure input drawing is projected within proper coordinate system
if (dwg.CoordinateSystem.Name <> "Transverse Mercator" or _
dwg.CoordinateSystem.Datum.Name <> "North American 1983 (mean for CONUS)") Then

doc.BatchUpdates = true

' Init the new coordinate system
dim coordSysNew
Set coordSysNew = Application.NewCoordinateSystem("Transverse Mercator")
coordSysNew.Datum = Application.NewDatum("North American 1983 (mean for CONUS)")
coordSysNew.Unit = Application.NewUnit("Meter")
coordSysNew.Parameters("centerLat").Value = 37.0 ' rough
coordSysNew.Parameters("centerLon").Value = -95.0 ' rough

' Fire up a coordinate converter and convert the input drawing to the proper
coordinate system
dim cc
Set cc = Application.NewCoordinateConverter()
cc.Prepare dwg.CoordinateSystem, coordSysNew

dim startTime
startTime = Now
If cc.ConvertsData() Then
Application.StatusText = "Converting to projection '" & coordSysNew.Name & "'...Please
wait"
Dim objIdx, obj

for objIdx = 0 to dwg.ObjectSet.Count - 1
set obj = dwg.ObjectSet(objIdx)
Application.StatusText = "Converting drawing to projection '" & coordSysNew.Name & "':
Projecting object " & objIdx & " of " & dwg.ObjectSet.Count
```

```
cc.Convert obj.Geom
next
End If


Application.MessageBox "Projection time: " & DateDiff("s", startTime, Now) & "
seconds.", "Reprojection Statistics"

' Attach the new coordinate system object to the document
dwg.CoordinateSystem = coordSysNew

doc.BatchUpdates = false
End If
End Sub
```

adamw at 1/23/2006 9:53 PM (#18172)

You could try shaving some time off by tuning the heaviest loop to:

```
'VBScript
objIdx = 1
objTotal = dwg.ObjectSet.Count
For Each obj in dwg.ObjectSet
  If objIdx Mod 10 = 0 Then
    Application.StatusText = "Projecting object " & objIdx & " of " & objTotal
  End If
  objIdx = objIdx + 1
  cc.Convert obj.Geom
Next
```

Convert Coordinate System via Code

Sent: Mon 01/23/2006 6:24 PM
To: Manifold List
From: Malcolm.Edwards@ingenix.com
Subject: Poor Performance With CoordinateConverter.convert()
http://lists.directionsmag.com/discussion/read.php?f=29&i=41391&t=41391

No answer but the code looks interesting. LAK

I'm new to the "new" forum and need some help. I' ve written a script which changes a drawing's coordinate system using the CoordinateConverter object. My VBA script (see below) works, but it is *magnitudes* slower than performing the same task manually using the GUI. For example, I have a fairly high point-detail drawing containing approximately 2200 objects. When I reproject the original drawing manually it takes roughly 7 seconds to finish. When I perform the same exact operation programmatically using the following script, it takes about 70 minutes!

Would someone look at the script below and let me know if I'm doing something wrong or can otherwise provide insight on how to achieve a higher level of performance.

Note, in my test the drawing being converted is in lat/long and I'm converting it to Transverse Mercator, NAD 1983 using the script.

Thanks in advance for any help you guys can offer!

```
Option Explicit

Sub Main
Dim doc, windows, window, dwg
set doc = Application.ActiveDocument

Set windows = Application.WindowSet

' ensure there is at least one opened window
If windows.Count < 1 Then
   Exit Sub
End If

Set window = windows.ActiveWindow

' ensure active window contains drawing
If window.ActiveComponent.Type <> ComponentDrawing Then
   Exit Sub
End If

set dwg = window.ActiveComponent

' Ensure input drawing is projected within proper coordinate system

If (dwg.CoordinateSystem.Name <> "Transverse Mercator" or _
       dwg.CoordinateSystem.Datum.Name <> _
      "North American 1983 (mean for CONUS)") Then
   doc.BatchUpdates = true

   ' Init the new coordinate system
   dim coordSysNew
   Set CoordSysNew = Application.NewCoordinateSystem("Transverse Mercator")
   CoordSysNew.Datum = Application.NewDatum("North American 1983 (mean for CONUS)")
   coordSysNew.Unit = Application.NewUnit("Meter")
   coordSysNew.Parameters("centerLat").Value = 37.0 ' rough
   coordSysNew.Parameters("centerLon").Value = -95.0 ' rough

' Fire up a coordinate converter and convert the input
' drawing to the proper coordinate system

   dim cc
   Set cc = Application.NewCoordinateConverter() cc.Prepare dwg.CoordinateSystem,
coordSysNew

dim startTime

startTime = Now

If cc.ConvertsData() Then
```

```
Application.StatusText = "Converting to projection '" & coordSysNew.Name & "'...Please
wait"

Dim objIdx, obj

for objIdx = 0 to dwg.ObjectSet.Count - 1

set obj = dwg.ObjectSet(objIdx)

Application.StatusText = "Converting drawing to projection '" & coordSysNew.Name & "':
Projecting object " & objIdx & " of " & dwg.ObjectSet.Count cc.Convert obj.Geom

next

End If

Application.MessageBox "Projection time: " & DateDiff("s", startTime,
Now) & " seconds.", "Reprojection Statistics"

' Attach the new coordinate system object to the document dwg.CoordinateSystem =
coordSysNew

doc.BatchUpdates = false

End If

End Sub
```

Tweaking a Surface Coordinate System

http://69.17.46.171/Site/Thread.aspx?id=19866&ti=632792261502970000

mdsumner at 3/28/2006 7:28 AM (#19947)

You can tweak the coordinate system of the surface directly - but be sure to isolate it from the original component - they behave like pointers.

Rough and ready, but this should get you going:

```
Sub Main
Set doc = Application.ActiveDocument
Set comps = doc.ComponentSet
Set comp = comps("Drawing")

Set cs = comp.CoordinateSystem
Set bbox = comp.ObjectSet.Box
xmin = bbox.Xmin - 10
xmax = bbox.Xmax + 10
ymin = bbox.Ymin -10
ymax = bbox.Ymax + 10

stepX = 20
stepY = 20
```

```
nx = round((xmax - xmin)/stepX) + 1
ny = round((ymax - ymin)/stepY) + 1

Set surf = doc.NewSurface("Surface", nx, ny, ValueTypeFloat64, cs, TRUE)

Set newCS = surf.CoordinateSystem
newcs.ParameterSet.Item(newcs.ParameterSet.ItemByName("localOffsetX")).Value = xmin
newcs.ParameterSet.Item(newcs.ParameterSet.ItemByName("localOffsetY")).Value = ymin
newcs.ParameterSet.Item(newcs.ParameterSet.ItemByName("localScaleX")).Value = stepX
newcs.ParameterSet.Item(newcs.ParameterSet.ItemByName("localScaleY")).Value = stepY
```

Verifying the Coordinate System for a Large Number of Components

Cannot retrieve the URL:
Many June and July 2006 postings seem to be lost at:
http://lists.directionsmag.com/discussion/list.php?f=29
and
http://lists.directionsmag.com/discussion/search.php?f=29

From: adamw@manifold.net
To: manifold-l@lists.directionsmag.com
Sent: Wed 06/28/2006 1:21 AM
Subject: RE: [Manifold-l] Verifying the Projection of a large number of Layers

> When I import a CAD drawing (DWG file) to Manifold
> there may be 40 to 50 layers. In Manifold, I have to verify the
> projection of each layer separately - even though all projections are
> the same, eg, Orthographic in Feet - very tedious. Is there a better
> way of doing this?

You can set the CoordinateSystemVerified property of the imported components using a script, eg (the following will only process drawings, you will probably want to extend this to process labels and also to report the names of the altered components):

```
Sub Main
  For Each c In Document.ComponentSet
    If c.Type = ComponentDrawing Then
      c.CoordinateSystemVerified = True
    End If
  Next
End Sub
```

Of course, you should only use this script when you are absolutely sure that the coordinate systems of the components are OK.

--
Adam Wachowski

Batch Reproject Drawings

http://69.17.46.171/Site/Thread.aspx?id=18933&ti=632754043299370000

grmapper at 2/13/2006 5:12 AM (#18960)

Thanks.

I updated the script to this working version based on the freestuff example.

One thing, this seems to take a long time. I have about 80 contour maps with millions of points. Since the reproject occurs on individual points within the drawings it requires significantly more processing time than simply doing it by hand.

Could this be done at a higher level than the Drawing.ObjectSet? There is a good chance I don't know what I am talking about, but I am learning. Here is the working script albeit a bit slow. Any suggestions on making my approach faster would be a big help.

Thanks again.

```
''ReProject all drawings in a folder to the new Target system
'<script language="VBScript">

Set app = Application

folderName = app.InputBox("Enter Data Folder","","Quads")

Set doc = Application.ActiveDocument
Set cmp = doc.ComponentSet(folderName)

Sub DoMerge(cmp)

If cmp.Type = ComponentDrawing Then

' Set Drawing = Components(DrawingIndex)

' -- create target coordinate system
Set Target = Application.NewCoordinateSystem("Universal Transverse Mercator - Zone 14
(N)")
Target.Datum = Application.NewDatum("North American 1983 (mean for CONUS)")
Target.Unit = Application.NewUnit("Meter")

' -- create coordinate system converter
Set Converter = Application.NewCoordinateConverter

' -- prepare coordinate converter
converter.Prepare cmp.CoordinateSystem, Target

Set Objects = cmp.ObjectSet
```

```
' -- project all objects within drawing
For IndexObj = 0 To Objects.Count-1
Set Object = Objects(IndexObj)
converter.Convert Object.Geom
Next

' -- modify coordinate system within drawing
cmp.CoordinateSystem = Target

ElseIf cmp.Type = ComponentFolder Then

For Each child In cmp.Children
DoMerge child

Next
End If
End Sub

Sub Main
    DoMerge Application.ActiveDocument.Componentset(folderName)
End Sub

'</script>
```

> I updated the script to this working version based on the freestuff
> example. One thing, this seems to take a long time. I have about 80
> contour maps with millions of points. Since the reproject occurs on
> individual points within the drawings it requires significantly more
> processing time than simply doing it by hand.
>
> Could this be done at a higher level than the Drawing.ObjectSet?

Not yet. This is in the wishlist for future editions.
Adam Wachowski

Build Coordinate System Object from WKT String

http://69.17.46.171/Site/Thread.aspx?id=28108&ti=632947930186770000

mdsumner on 9/25/2006 2:56 PM (#28109)

```
Sub Main
Set comps = Document.ComponentSet

Set cs = App
ication.NewCoordinateSystemFromTextWKT("PROJCS[""unnamed"",GEOGCS[""GCS_WGS_1984"",DAT
UM[""D_WGS_1984"",SPHEROID[""WGS_1984"",6378137.000000,298.257224]],PRIMEM[""Greenwich
```

```
"",0.0],UNIT[""Degree"",0.0174532925199433]],PROJECTION[""Orthographic""],PARAMETER[""
Fa
se_Easting"",0.0],PARAMETER[""Fa
se_Northing"",0.0],PARAMETER[""
ongitude_Of_Center"",0.0],PARAMETER[""
atitude_Of_Center"",0.0],UNIT[""Meter"",1.0]]")

End Sub
```

Convert a Point from Drawing Coordinate System to Lat/Lon

http://69.17.46.171/Site/Thread.aspx?id=166&ti=632079515400000000

adamw on 11/25/2003 8:00 AM (#213)

```
' Here is the example code that converts a point
' from drawing coordinate system to lat/lon.

{code - temp tag to be replaced}

Option Explicit

Sub Main
  Dim Pt, Drwg, Converter

  Set Pt = Application.NewPoint(1000, 1000)

  Set Drwg = Document.ComponentSet("Drawing")

  Set Converter = Application.NewCoordinateConverter()

  Converter.Prepare Drwg.CoordinateSystem, _
     Application.DefaultCoordinateSystemLatLon

   ' Following line converts (1000, 1000) from drawing
   ' coordinate system to lat/lon.
  Converter.Convert Pt

  Application.Messagebox Pt.X & ":" & Pt.Y

End Sub
{/code - temp tag to be replaced}


' Goal is to create a bounding rectangle for each component and store it in a lat/long
' drawing component.
'
' Speaking of drawings, you can project each drawing object to lat/lon and compute
' common bounding box of the projected objects.
'
' Speaking of images and surfaces, while you can project each pixel to lat/lon and
' compute common bounding box of the projected pixels, it might be better to project
' a, say, 10x10 grid that covers the entire component and compute a bounding box of
```

```
' that area (in many cases, projecting  just the corners will not work).
'
' Case "majorAxis", "eccentricity", "centerX", "centerY", "centerZ"
'
' ignore these cause I don't have a clue as to what they do
'
' These are the datum:
' ~~~~~~~~~~~~~~~~~~
' parameters.majorAxis          = size of the major axis in meters.
' eccentricity                  = square root from (1- (minor*minor)/(major*major))
' centerX, centerY, and centerZ = offsets to the center of the ellipsoid
                                  relative to WGS 84 in meters.
```

Extract Map Layer Coordinate System Information to Table Columns

```
' ****************************************************************
' *     THE FOLLOWING MANIFOLD-L THREAD GIVES THE BACKGROUND.    *
' *          THIS IS ADAM'S SUGGESTION IN A SCRIPTED FORM.       *
' *                   L. Ketch - April 9, 2006                   *
' ****************************************************************
'
' Sent:    Thu 04/06/2006 5:19 PM
' From:    David Weinschrott [david@prodevassociates.com]
' To:      manifold-l@lists.directionsmag.com
' Subject: [Manifold-l] Finding the bad apple
'
' I have a map I have been adding stuff to for a couple of weeks. All of a
' sudden things have slowed down a lot. My guess is that one of those layers
' has an odd projection.  In my understand, In order to find the bad apple I
' have to find each of those layers in the project mnanager and check its
' projection. Is there any other quicker way -- it would be cool to be able
' to put up a table that would list projections of each layer.  One could have
' a button that would change all layers to a selected projection, but that is
' a bit drastic when you don't know which one might need specific projection
' attention.
'
' David J. Weinschrott, PhD
'
' ----------------------------------------------------------------------
'
' Sent:    Fri 04/07/2006 10:06 AM
' From:    adamw@manifold.net
' To:      manifold-l@lists.directionsmag.com
' Subject: RE: [Manifold-l] Finding the bad apple
'
' > ... it would be cool to be able to put up a table that
' > would list projections of each layer.
'
' Here is what you could do:
'
' Create a table named "Data" with a text column named "Name". Open the table
' and add a record for each component you are interested in. Create a new query
' and set its text to:
'
```

```
'  SELECT [Name], CoordSys([Name] AS COMPONENT) [CoordSys] INTO [CoordSystems] FROM
[Data];
'
' Run the query. Open the resulting table. Right click the "CoordSys" column,
' select Format, set formatting style to "XML" and click OK. Resize the column.
' You should see the name of the coordinate system preset in the first 50 or
' so characters of the XML (between <name> and </name>).
'
' --
' Adam Wachowski
' Manifold Development Team
' ************************************************************************

Option Explicit

Sub Main()

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

  Dim Comp
  Dim CoordSys_Table
  Dim Cols, Col
  Dim Rcrd, Rcrds
  Dim Active_Window
  Dim Coord_Sys, CoordSys_ParmSet
  Dim Map_Layer, Map_Layers

  Set Active_Window = Application.WindowSet.ActiveWindow

  If Active_Window.Component.Type <> ComponentMap Then
      Application.MessageBox "The Active Window must be a Map"
      Exit Sub
  End If

  Set Map_Layers = Active_Window.Component.LayerSet

  Delete_Component("Layer_CoordSystems")
  Set CoordSys_Table = Document.NewTable("Layer_CoordSystems")
  Set Cols = CoordSys_Table.ColumnSet

    ' NOTE: following 4 lines do not work if run under the Debugger.
    ' The column name is not changed and the type is not set. You
    ' get an error later when the code trys to find column "MapLayer"
    ' Code works fine if not run through the Debugger.
  Set Col  = Cols.Item(0)
  Col.Name = "MapLayer"
  Col.Type = ColumnTypeWText
  Col.Size = 100

  Set Col  = Cols.NewColumn
  Col.Name = "CoordSys_Name"
  Col.Type = ColumnTypeWText
  Col.Size = 100
  Cols.Add(Col)

  Set Col  = Cols.NewColumn
```

```
Col.Name = "Datum"
Col.Type = ColumnTypeWText
Col.Size = 100
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "EllipsoidName"
Col.Type = ColumnTypeWText
Col.Size = 100
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "CoordSys_Unit"
Col.Type = ColumnTypeWText
Col.Size = 50
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "localOffsetX"
Col.Type = ColumnTypeFloat32
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "localOffsetY"
Col.Type = ColumnTypeFloat32
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "localScaleX"
Col.Type = ColumnTypeFloat64
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "localScaleY"
Col.Type = ColumnTypeFloat64
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "ScaleCorrectionX"
Col.Type = ColumnTypeFloat64
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "ScaleCorrectionY"
Col.Type = ColumnTypeFloat64
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "FalseEasting"
Col.Type = ColumnTypeFloat64
Cols.Add(Col)

Set Col  = Cols.NewColumn
Col.Name = "FalseNorthing"
Col.Type = ColumnTypeFloat64
Cols.Add(Col)
```

```
  Set Col  = Cols.NewColumn
  Col.Name = "majorAxis"
  Col.Type = ColumnTypeFloat64
  Cols.Add(Col)

  Set Col  = Cols.NewColumn
  Col.Name = "Eccentricity"
  Col.Type = ColumnTypeFloat64
  Cols.Add(Col)

  Set Col  = Cols.NewColumn
  Col.Name = "CoordSys_XML"
  Col.Type = ColumnTypeWText
  Col.Size = 2000  ' >1000 required here or the script fails
  Cols.Add(Col)

  Set Rcrds = CoordSys_Table.RecordSet

  For Each Map_Layer in Map_Layers
     Set Comp = Map_Layer.Component
     Set Coord_Sys = Comp.CoordinateSystem
     Set CoordSys_ParmSet = Coord_Sys.ParameterSet

     Rcrds.AddNew()
     Set Rcrd = Rcrds.LastAdded

     Rcrd.Data("MapLayer") =  Comp.Name

     Rcrd.Data("CoordSys_Name") =  Coord_Sys.Name
     Rcrd.Data("Datum")         =  Coord_Sys.Datum.Name
     Rcrd.Data("EllipsoidName") =  Coord_Sys.Datum.Ellipsoid.Name
     Rcrd.Data("CoordSys_Unit") =  Coord_Sys.Unit.Name

     Rcrd.Data("localOffsetX")  =  CoordSys_ParmSet.Item("localOffsetX").Value
     Rcrd.Data("localOffsetY")  =  CoordSys_ParmSet.Item("localOffsetY").Value

     Rcrd.Data("localScaleX")  =  CoordSys_ParmSet.Item("localScaleX").Value
     Rcrd.Data("localScaleY")  =  CoordSys_ParmSet.Item("localScaleY").Value

     Rcrd.Data("ScaleCorrectionX")  =  CoordSys_ParmSet.Item("ScaleX").Value
     Rcrd.Data("ScaleCorrectionY")  =  CoordSys_ParmSet.Item("ScaleY").Value

     Rcrd.Data("FalseEasting")  =  CoordSys_ParmSet.Item("FalseEasting").Value
     Rcrd.Data("FalseNorthing")  =  CoordSys_ParmSet.Item("FalseNorthing").Value

     Rcrd.Data("majorAxis")  =  CoordSys_ParmSet.Item("majorAxis").Value
     Rcrd.Data("Eccentricity")  =  CoordSys_ParmSet.Item("Eccentricity").Value

     Rcrd.Data("CoordSys_XML")  =  Coord_Sys.ToXML()
  Next

  CoordSys_Table.Open()

End Sub

' ************************************************************************
```

Terrain Fly-Through

http://69.17.46.171/Site/Thread.aspx?id=6062&ti=632507946600000000

mdsumner on 11/1/2004 3:05 AM (#6063)

Hello, this is a bit of fun that is now possible with 6.00 SP1

This script will take a terrain that you have in Manifold, and fly you along based on a drawing of line segments - the smoothness of your ride is basically dependent upon how you construct the lines - the script takes each line segment - moves to its center point and aligns the camera angle to be looking in the direction the line is pointing (so simple, single branch lines with 2 points are assumed).

There is some basic trig that works out the right values for the camera angles from the point to point bearing, I just guessed until I got it right really - so no guarantees or anything, but C is for cookie, that's good enough for me. It could use the surface slope to pick a sensible camera Z angle I guess, but for now I keep my eyes to the road.

The script currently assumes that you have "Height" values in your drawing for each line segment - so you can add an offset at the start of the script to fly above the ground some specified amount.

It is set to expect your terrain window to be open and active, that there is a drawing "Drawing" with line segments, and that each line segment has an attribute height. Easy to turn this off if desired.

There is also a delay funtion, to make Manifold wait a specified number of seconds between each move - I suggest you start with small values like 0.1 seconds if you want to try this.

Enjoy, I hope it's of interest to someone. The Manifold object model is really coming along . . .

I might even post up an example .map file to make this easier for people to see in operation.

Note that it would be easy to export each terrain view to image files that could then be animated.

```
'##**************************************************************************

'## TerrainFlyThrough - illustration of scripted terrain fly through
'## Author: Michael Sumner Date: November 1 2004 Version: 1.0.0
'## mdsumner@utas.edu.au
'##
'## Platform: Windows XP 2002 SP2.
'## Manifold 6.0 SP1,
'##
'## Language: VBScript (Manifold System)
```

```
'##
'## Permission to use, modify and distribute this code is hereby granted,
'## providing this entire comment section is included. No warranty is
'## expressed or implied.
'##

Sub Main

     ' name of drawing component to look for
   name = "Drawing"

     ' name of column in drawing's table containing line heights
     ' THIS IS ASSUMED to exist
   htColumnName = "Height"

     ' choose the z direction of the camera (looking down a bit)
   dZ = -2

     ' how high above the surface?
   zOffset = 0

     ' the number of seconds to wait between each move (use small values like 0.1)
   waitThisLong = 0

   Set doc = Application.ActiveDocument
   Set comps = doc.ComponentSet

   'obtain active terrain window

   Set wset = Application.WindowSet
   Set win = wset.ActiveWindow

   If win.Component.Type <> ComponentTerrain Then
      msgbox "Active window is not a terrain, exiting"
      Exit Sub
   End If


   ' obtain drawing containing the line segments

   dwgID = comps.ItemByName(name)

   If dwgID < 0 Then
      msgbox "No component named " & name
   End If

   Set dwg = comps.Item(dwgID)

     ' obtain the recordSet for the height values
   Set records = dwg.OwnedTable.RecordSet

     ' obtain the line segments from the drawing
   Set dwgObjs = dwg.ObjectSet

     'msgbox win.CameraDX & vbcrlf & win.CameraDY & vbcrlf & win.CameraDZ
     'exit sub
```

```
     'iterate over all objects

   For n = 1 To dwgObjs.Count
      Set obj = dwgObjs.Item(n-1)
        ' act only on line objects (using first two points of first branch)

      If obj.Type = ObjectLine Then
          Set ptSet = obj.Geom.BranchSet.Item(0).PointSet
            ' obtain the two points, and calculate camera directions
            ' from the bearing from pt1 to pt2
            ' (these range from -10 to 10 for all 3 dimensions - not totally
            ' sure this is right but it seems fine)

          Set pt1 = ptSet(0)
          Set pt2 = ptSet(1)
          bearing = pt1.BearingTo(pt2)

          win.CameraDY = cos(bearing)*10
          win.CameraDX = sin(bearing-180)*10
          win.CameraDZ = dZ


          '' ###################################################################
          '' ## turn this section off if you have no height data in your drawing
          ' obtain the height value from the surface (this is mostly so you can
          ' make it more than zero if desired

             win.CameraHeight = records(records.ItemByID(obj.ID)).Data(htColumnName) _
                              + zOffset
          '' ##
          '' ###################################################################

            ' create a new point and move the window view to it, and refresh

          Set nPt = Application.NewPoint
          nPt.X = obj.Geom.Center.X
          nPt.Y = obj.Geom.Center.Y
          win.MoveToLocation nPt, FALSE
          win.Refresh()

            'waste some time?
          delay waitThisLong

            'exit sub

   End If

   Next

End Sub


' a function to waste the specified number of seconds

Function delay(wait)

     'Delay redirect for two seconds'
```

```
   Dim StartTime, EndTime, WaitTime

     ' Get Start Time'
   StartTime = Timer()

     'Get End Time'
   EndTime = Timer()

     'Determine how long it took'
   WaitTime = EndTime - StartTime

     'Continue checking the elapsed time until it reaches "wait" seconds'

   do while WaitTime < wait
         'Get End Time'
       EndTime = Timer()
         'Determine how long it took'
       WaitTime = EndTime - StartTime
   loop

End Function
```

Mikes code with Option Explicit added and minor other changes.
http://forum.manifold.net/Site/Thread.aspx?id=31217&ti=633004906914170000

```
Option Explicit

'#####################################################################
'## TerrainFlyThrough - illustration of scripted terrain fly through
'## Author: Michael Sumner Date: November 1 2004 Version: 1.0.0
'## mdsumner@utas.edu.au
'##
'## Platform: Windows XP 2002 SP2.
'## Manifold 6.0 SP1,
'##
'## Language: VBScript (Manifold System)
'##
'## Permission to use, modify and distribute this code is hereby
'## granted, providing this entire comment section is included.
'## No warranty is expressed or implied.
'#####################################################################

'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
'$$
'$$     THIS IS MIKE'S GREAT CODE WITH A FEW UPDATES FOR V7
'$$     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'$$
'$$   Option Explicit Added
'$$   V7 properties and methods.
'$$     Examples:
'$$       Application.Messagebox            REPLACES MsgBox
'$$       Set Comps = Document.ComponentSet REPLACES Set Comps = _
'$$       Application.ActiveDocument.ComponentSet
'$$   L.Ketch - Nov 2006
'$$
```

```
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Sub Main
   Dim Comps
   Dim Drwg_Name, htColumnName
   Dim dZ, zOffset, waitThisLong
   Dim wSet, Win, DrwgID, Drwg
   Dim Rcrds, DrwgObjs, Obj
   Dim ptSet, Pt1, Pt2, Bearing, nPt, n
   Dim WScript, NumberOfObjs

   ' IMPORTANT: Make sure the Drawing containing the fly-through track lie
   ' has exactly the same projection parametres as the surface/terrain.

   ' //// SET THE FOLLOWING FIVE PROGRAM PARAMETRES \\\\

      ' choose the z direction of the camera (looking down a bit)
   dZ = -4
      ' how high above the surface?
   zOffset = 100
      ' the number of seconds to wait between each move (use small values like 0.1)
   waitThisLong = .1
     ' Drwg_Name of drawing component to look for
  ' Drwg_Name = "Track2_492segments"
   Drwg_Name = "Drawing_1179segments"

     ' Drwg_Name of column in drawing's table containing line heights
     ' THIS IS ASSUMED to exist
   htColumnName = "Height"
   ' ////           END - PROGRAM PARAMETRES        \\\\

   Set WScript = CreateObject("WScript.Shell")
   Set Comps = Document.ComponentSet

      'obtain active terrain window
   Set wSet = Application.WindowSet
   Set Win  = wSet.ActiveWindow

   If Win.Component.Type <> ComponentTerrain Then
      Application.Messagebox "Active window is not a terrain, exiting"
      Exit Sub
   End If

      ' obtain drawing containing the line segments
   DrwgID = Comps.ItemByName(Drwg_Name)

   If DrwgID < 0 Then
      Application.Messagebox  "No component named " & Drwg_Name
   End If


   Set Drwg = Comps.Item(DrwgID)

      ' Obtain the recordSet for the height values
   Set Rcrds = Drwg.OwnedTable.RecordSet
      ' obtain the line segments from the drawing
   Set DrwgObjs = Drwg.ObjectSet
```

```
      ' Iterate over all objects
  NumberOfObjs = DrwgObjs.Count
   For n = 1 To NumberOfObjs

      Application.StatusText = "Frame " & cStr(n) & " of " & cStr(NumberOfObjs)

      Set Obj = DrwgObjs.Item(n-1)
         ' act only on line objects (using first two points of first branch)
      If Obj.Type = ObjectLine Then
         Set ptSet = Obj.Geom.BranchSet.Item(0).PointSet
            ' obtain the two points, and calculate camera directions
            ' from the bearing from pt1 to pt2
            ' (these range from -10 to 10 for all 3 dimensions - not totally sure this
is
            ' right but it seems fine)
         Set Pt1 = ptSet(0)
         Set Pt2 = ptSet(1)
         Bearing = Pt1.BearingTo(Pt2)
         Win.CameraDY = Cos(Bearing) * 10
         Win.CameraDX = Sin(Bearing) * 10
'         Win.CameraDX = Sin(Bearing - 180) * 10
         Win.CameraDZ = dZ

         '' ######################################################
         '' ## Turn this section off if you have no height data
         '' ## in your drawing. Obtain the height value from the
         '' ## surface (this is mostly so you can make it more
         '' ## than zero if desired.
         Win.CameraHeight = Rcrds(Rcrds.ItemByID(Obj.ID)).Data(htColumnName) + zOffset
         '' ######################################################

            ' Create a new point and move the window view to it,
            ' and refresh.
         Set nPt = Application.NewPoint
         nPt.X = Obj.Geom.Center.X
         nPt.Y = Obj.Geom.Center.Y

         Win.MoveToLocation nPt, FALSE

         Win.Refresh()

            ' waste some time?
            ' Delay waitThisLong
         If waitThisLong > 0 Then Delay(waitThisLong)

      End If

   Next

End Sub

' ****************************************************************

Function Delay(Wait)
   ' A function to waste the specified number of seconds
  Dim StartTime, EndTime, WaitTime
```

```
    ' Get Start Time'
  StartTime = Timer()
    ' Get End Time'
  EndTime = Timer()
    ' Determine how long it took'
  WaitTime = EndTime - StartTime

    ' Continue checking the elapsed time until it reaches "wait" seconds
  Do While WaitTime < Wait
      ' Get End Time'
    EndTime = Timer()
      ' Determine how long it took'
    WaitTime = EndTime - StartTime
  Loop

End Function

' ****************************************************************
```

Terrain Fly-Through: Save the X, Y and Z Flight Coordinates

From: Michael Sumner

Sent: December 11, 2006 3:23 PM
To: Dr. Harald von der Osten
Cc: Manifold-L
Subject: Re: [Manifold-l] saving fly-through trajectory

> http://lists.directionsmag.com/discussion/read.php?f=29&i=42884&loc=0&t=42884
> just an idea...: is it possible to save the (x,y,z)-coordinates of the
> camera movement, forced by striking the keys (Q, E, ...) to a file?
> These trajectory could then be smoothed and could be the input for the
> flying-through script...

That is not currently possible AFAIK. You could do it in a slower more manual way by navigating as normal and then at each desired "main point" capture the current coordinates with a script and archive them.

Here's a basic example, you would modify this to write the required values to a text component or table, then manually step through your desired navigation, running the script once each move, process the trajectory and then input to the other script:

```
Sub Main

   Set Comps = Document.ComponentSet

      'obtain active terrain window
   Set wSet = Application.WindowSet
   Set Win  = wSet.ActiveWindow

   If Win.Component.Type <> ComponentTerrain Then
```

```
      Application.Messagebox "Active window is not a terrain, exiting"
      Exit Sub
   End If

      ' get the current location
    Set pt = Win.Location ' or LocationLatLon if required
    mess = ""
      ' append the data to a string
    mess = mess & "point: " & pt.X & " , " & pt.Y & vbcrlf

      '  direction of camera
      ' (may not be necessary, depending on how you process the locations)
    mess = mess & "cameraDX: " & Win.CameraDX & vbcrlf
    mess = mess & "cameraDY: " & Win.CameraDY & vbcrlf
    mess = mess & "cameraDZ: " & Win.CameraDZ & vbcrlf


    Application.Messagebox mess

End Sub
```

HTH, Mike

From: Frank O'Donnell
Sent: December 12, 2006 10:33 PM
To: manifold-l@lists.directionsmag.com
Subject: [Manifold-l] Navigating within a 3D terrain view
http://lists.directionsmag.com/discussion/read.php?f=29&i=42895&loc=0&t=42895

..... when I open the terrain associated with the surface, I find myself quite far away from the data I want to see. The data is centered at approximately 33 44.109 N 118 24.095 W, whereas when the 3D view initializes it always starts at 33 42.571 N 118 21.035 W. I've printed out the section "Keyboard Navigation" from the Manifold help file, but no combination of strafing/tilting/rotation that I'm finding is getting me toward the data in anything at a faster than glacial pace. It would be convenient if I could, say, enter lat-lon values to fly to immediately, but I'm also not seeing a way to do that.

So I have two questions. First, why does the 3D view initialize at that particular point relatively distant from the data? I'm wondering if perhaps that was the center of the larger set of surface data that I trimmed away? (In the deletion process, I told Manifold not to hold on to the old data.) Is there a way to reset the data center so that a 3D terrain view initializes above the remaining data? And second, if I start where the 3D terrain view comes up now, is there a much faster way of getting to the data I want to see? Thanks for any help.

From: jburn_gis@cogeco.ca
Sent: December 12, 2006 11:13 PM
To: Frank O'Donnell; manifold-l@lists.directionsmag.com
Subject: re: [Manifold-l] Navigating within a 3D terrain view
http://lists.directionsmag.com/discussion/read.php?f=29&i=42896&t=42895

Something to try:
Open your map (or surface) so that you can see both your 3D terrain, and 2D surface. Now open the "world" pane (alt+sft+"o"). You can use whatever combination you like, but I like to show the track on my 2D view, and center on my 3D view. Now, as you move around the terrain, you'll see that the crosshair move on the 2D view. You can now use the regular "pan" function in 2D to move your area of interest towards the crosshairs, which will also change in your 3D view.

Happy flying;-)

From: Michael Sumner
Sent: December 12, 2006 11:36 PM
To: Frank O'Donnell
Cc: manifold-l@lists.directionsmag.com
Subject: Re: [Manifold-l] Navigating within a 3D terrain view
http://lists.directionsmag.com/discussion/read.php?f=29&i=42897&t=42895

> Is there a way to reset the data center so that a 3D terrain view initializes above the remaining
> data? And second, if I start where the 3D terrain view comes up now, is there a much faster
> way of getting to the data I want to see?

2. Using the World Pane you can use another component to centre the view

Surface Analysis Script – slope, orientation, vertical & horizontal curvature calculation

http://www.georeference.org/Forums/tabid/71/forumid/7/tpage/1/view/topic/postid/353/Default.aspx#375
http://69.17.46.171/Site/Thread.aspx?id=362&ti=632526961800000000

Abram
12/02/2003 12:45 PM

This small script allow the user to perform specific analysis (slope, orientation, vertical and horizontal curvatures). This analysis is performed using the methods described by Burrough and allows the user to defined the window size (nxn). This was developed in order to manage lidar digital elevation data with 1 meter resolution, but can be used with any dem resolution. the power of user defined window is the management of the context encountered in the analysis.

The code is open to any use, but be careful, due to the scripting version it is **slow.**

The resulting surfaces are appearing black because the display setup is currently not accessible through script (but should appear in the next version).

Feel free to contact me for any comment or additional information.

abram

```
' *** DEM ANALYSIS SCRIPT ***
' **************************
' *** performs slope, orientation, vertical and horizontal curvature calculation
' *** based on the Burrough(1998) description
' *** allows user to define the size of the window (nxn) (n = odd number!)
' *** version 0.2 (December 3rd 2003)
' *** abram.pointet(at)epfl.ch
' *** http://lasig.epfl.ch
' ********************************************************************************
' *** 3X3 moving window description ***
' *** **********
' *** *Z1*Z2*Z3*
' *** **********
' *** *Z4*Z5*Z6*
' *** **********
' *** *Z7*Z8*Z9*
' *** **********
' ********************************************************************************
' *** 5X5 moving window description ***
' *** ****************
' *** *Z1* *Z2* *Z3*
' *** ****************
' *** * * * * * *
' *** ****************
' *** *Z4* *Z5* *Z6*
' *** ****************
' *** * * * * * *
' *** ****************
' *** *Z7* *Z8* *Z9*
' *** ****************

Sub Main
   ' *** USER DEEFINED PARAMETERS ***
   ' set moving window parameter (odd number!)
   windowsize = int(InputBox("Enter window size (odd integer!)","Moving window size"))

   ' calculate slope
   If (Application.Messagebox ("Do you want to calculate the slopes ?", vbYesNo +
vbQuestion, "Slope") = vbYes) Then
      cs = true
   Else
      cs = false
   End If

   ' calculate orientation
   If (Application.Messagebox ("Do you want to calculate the orientation ?", vbYesNo +
vbQuestion, "Orientation") = vbYes) Then
      co = true
   Else
      co = false
   End If

   ' calculate vertical curvature
```

```
   If (Application.Messagebox ("Do you want to calculate the vertical curvature ?",
vbYesNo + vbQuestion, "Vertical Curvature") = vbYes) Then
      cvc = true
   Else
      cvc = false
   End If

   ' calculate horizontal curvature
   If (Application.Messagebox ("Do you want to calculate the horizontal curvature ?",
vbYesNo + vbQuestion, "Horizontal Curvature") = vbYes) Then
      chc = true
   Else
      chc = false
   End If


   ' *** END OF USER DEFINED PARAMETERS ***
   ' check parameters
   If Not CBool(windowsize Mod 2) Then
      Application.Messagebox "Window size has to be set using odd numbers!"
      Exit Sub
   End If


   ' retrieve dem component & internal parameters
   Set window = Application.WindowSet.ActiveWindow
   Set component = window.Component

   If window.Component.Type <> ComponentSurface Then
     Application.Messagebox "No active surface."
     Exit Sub
   End If

   Set surface = window.Component
   tot_cols = surface.Width
   tot_rows = surface.Height
   surface_name = surface.Name
   set surface_coordsyst = surface.CoordinateSystem
   set coordparameters = surface_coordsyst.Parameters
   set resolution_param = surface.CoordinateSystem.Parameters.Item(6)
   surface_resolution = resolution_param.Value
   R = surface_resolution
   Set surface_pxset = surface.PixElset


   ' offset calculation
   offset = windowsize/2-0.5
   If offset > (tot_cols/4) or offset > (tot_rows/4) then
     Application.Messagebox ("Window size exceding half domain!")
     Exit Sub
   End If

   ' set parent folder for new analysis surfaces
   Set document = Application.DocumentSet.Item(0)
   Set folder = document.NewFolder(surface_name & "_analysis", false)

   ' create new surfaces with the same characteristics as the dem
```

```
   If cs = true then
      Set slope = document.NewSurface(surface_name & "_Slope " & windowsize & "x" &
windowsize, tot_cols, tot_rows, ValueTypeFloat64, surface_coordsyst, 0)
      slope.Folder = folder
      Set slope_pxset = slope.PixElset
   End If

   If co = true then
      Set orientation = document.NewSurface(surface_name & "_Orientation " &
windowsize & "x" & windowsize, tot_cols, tot_rows, ValueTypeFloat64,
surface_coordsyst, 0)
      orientation.Folder = folder
      Set orientation_pxset = orientation.PixElset
   End If

   If cvc = true then
      Set vertcurv = document.NewSurface(surface_name & "_Vertical Curvature " &
windowsize & "x" & windowsize, tot_cols, tot_rows, ValueTypeFloat64,
surface_coordsyst, 0)
      vertcurv.Folder = folder
      Set vertcurv_pxset = vertcurv.PixElset
   End If

   If chc = true then
      Set horcurv = document.NewSurface(surface_name & "_Horizontal Curvature " &
windowsize & "x" & windowsize, tot_cols, tot_rows, ValueTypeFloat64,
surface_coordsyst, 0)
      horcurv.Folder = folder
      Set horcurv_pxset = horcurv.PixElset
   End If

   ' calculate border effect area
   start_row = offset
   end_row = tot_rows-offset-1
   start_col = offset
   end_col = tot_cols-offset-1

   If cs = true or co = true or cvc = true or chc = true then
     ' manage loops across the surface

      For rowcount = start_row to end_row

         For colcount = start_col to end_col

            Set Z5 = surface_pxset.item(surface_pxset.ItemByXY(colcount,rowcount))

            Set Z2 = surface_pxset.item(surface_pxset.ItemByXY(colcount,rowcount-
offset))
            Set Z8 =
surface_pxset.item(surface_pxset.ItemByXY(colcount,rowcount+offset))
            Set Z4 = surface_pxset.item(surface_pxset.ItemByXY(colcount-
offset,rowcount))
            Set Z6 =
surface_pxset.item(surface_pxset.ItemByXY(colcount+offset,rowcount))


            If cvc = true or chc = true then
```

```
               Set Z1 = surface_pxset.item(surface_pxset.ItemByXY(colcount-
offset,rowcount-offset))
               Set Z3 =
surface_pxset.item(surface_pxset.ItemByXY(colcount+offset,rowcount-offset))
               Set Z7 = surface_pxset.item(surface_pxset.ItemByXY(colcount-
offset,rowcount+offset))
               Set Z9 = surface_pxset.item(surface_pxset.ItemByXY(colcount-
offset,rowcount+offset))
            End If

            If cvc = true then
               Set VC5 =
vertcurv_pxset.item(vertcurv_pxset.ItemByXY(colcount,rowcount))
            End If

            If chc = true then
               Set HC5 = horcurv_pxset.item(horcurv_pxset.ItemByXY(colcount,rowcount))
            End If

            'calculate base values
            If cvc = true or chc = true then
               D = ((Z4.Value+Z6.Value)/2-Z5.Value)/(R*R)
               E = ((Z2.Value+Z8.Value)/2-Z5.Value)/(R*R)
               F = (-Z1.Value+Z3.Value+Z7.Value-Z9.Value)/(4*R*R)
            End If

            G = (-Z4.Value+Z6.Value)/(2*R)
            H = (Z2.Value-Z8.Value)/(2*R)

            If cs = true then
                ' Slope calculation
               Set S5 = slope_pxset.item(slope_pxset.ItemByXY(colcount,rowcount))
               Slope_value = atn(sqr(G*G+H*H))*180/3.14159265359
               S5.Value = Slope_value
            End If

            If co = true then

               ' Orientation calculation
               If G=0 then
                  If H > 0 then
                     Orientation_value = 180
                  ElseIf H < 0 then
                     Orientation_value = 0
                  Else
                     Orientation_value = -1
                  End If
               Else
                  Orientation_value = Abs(Atn(-H/-G)*180/3.14159265359)
                  If G < 0 and H < 0 then
                     Orientation_value = Orientation_value
                  End If

                  If G < 0 and H > 0 then
                     Orientation_value = Orientation_value + 90
                  End If
```

```
                  If G > 0 and H < 0 then
                     Orientation_value = Orientation_value + 270
                  End If

                  If G > 0 and H > 0 then
                     Orientation_value = Orientation_value + 180
                  End If
               End If

               Set O5 =
orientation_pxset.item(orientation_pxset.ItemByXY(colcount,rowcount))
               O5.Value = Orientation_value

            End If

            If cvc = true or chc = true then
                 ' Vertical and horizontal curvature calculation
               If G=0 and H=0 then
                  Vertcurv_value = 0
                  Horcurv_value = 0
               Else
                  Vertcurv_value = 2*(D*G*G+E*H*H+F*G*H)/(G*G+H*H)
                  Horcurv_value = -2*(D*H*H+E*G*G-F*G*H)/(G*G+H*H)
               End If

               If cvc = true then
                  VC5.Value = Vertcurv_value
               End If

               If chc = true then
                  HC5.Value = Horcurv_value
               End If
            End If

         Next
      Next

   End If

End Sub
```

Extracting a Surface Height at a given Latitude/Longitude

AdamW
08/17/2004 9:06 AM
http://www.georeference.org/Forums/tabid/71/forumid/5/tpage/1/view/topic/postid/3194/Default.aspx#3710

Here is the script which obtains the height of the surface at the hard-coded lat/lon location (VBScript):

```
Option Explicit

Sub Main
  Dim Cmps, Srf, Conv, Cssrc, Cstgt
```

```
  Dim Pt, Pixs, Pind
  Dim xco, yco
  Dim X, Y, Z
  Dim MinX, MaxX, MinY, MaxY

  Set Cmps = Application.ActiveDocument.ComponentSet
  Set Srf = Cmps("Halifax_CentralNS Surface")

     ' create coordinate converter
  Set Conv = Application.NewCoordinateConverter()
  Set Cssrc = Application.DefaultCoordinateSystemLatLon ' <-- lat/lon
  Set Cstgt = srf.CoordinateSystem

  Conv.Prepare Cssrc, Cstgt ' <-- lat/lon to surface

     ' get cursor coordinates (near the point in the TargetPoint drawing)
  xco = -64.2
  yco = 44.8
  Set Pt = Application.NewPoint(xco, yco)
  Conv.Convert Pt

     ' obtain Z
  Set Pixs = Srf.PixelSet


     ' Code between dashes has been added to Adam's original to prevent
     ' a Manifold error message when one attempts to return height
     ' information for a point not contained in the surface. L. Ketch
     ' ----------------------------------------------------
  MinX = Pixs.box.xMin
  MaxX = Pixs.box.xMax
  MinY = Pixs.box.yMin
  MaxY = Pixs.box.yMax

  X = pt.X
  Y = srf.Height - pt.Y - 1

  If X < MinX or X > MaxX or Y < MinY or Y > MaxY Then
     Application.MessageBox "Point is Not on the Surface (X = " & xco & "  y = " & yco
& ")"
     Exit Sub
  End If

     ' ----------------------------------------------------

     ' Line below will generate an error if the point xco, yco does not exist
     ' on the surface ... circumvented by code above - LAK

  Pind = pixs.ItemByXY(X, Y)

  If Pind > 0 Then
     Z = pixs(pind).Value
     Application.MessageBox "Height = " & Z
  End If

End Sub
```

Batch Transform Image Gamma

http://69.17.46.171/Site/Thread.aspx?id=29027&ti=632975122283300000

ColinD on 10/18/2006 2:26 PM (#29028)

Does anyone have a script that would get me started on batch image transforms such as altering Gamma by a fixed amount?

mdsumner on 10/26/2006 9:13 PM (#29384)

' VBScript:

```
Sub Main
  Set doc = Application.ActiveDocument
  Set comps = doc.ComponentSet
  Set comp = comps("Image")

  Set anlz = doc.NewAnalyzer

  anlz.Gamma comp.PixelSet, 40
End Sub
```

If you want to modify a selection:

```
Sub Main
  Set doc = Application.ActiveDocument
  Set comps = doc.ComponentSet
  Set comp = comps("Image")

  Set anlz = doc.NewAnalyzer

  anlz.Gamma comp.Selection, 40
End Sub
```

To loop over every image in the project:

```
Sub Main
  Set doc = Application.ActiveDocument
  Set comps = doc.ComponentSet
  Set anlz = doc.NewAnalyzer

  For each comp in comps
     If comp.TypeName = "Image" Then anlz.Gamma comp.PixelSet, 40
  Next
End Sub
```

Sum all Surfaces in a Project

nielsraes on 12/4/2006 8:11 AM (#31425)

Dear List,

I have to sum all surfaces in a many different maps. I can use the Transform dialog, but that is going to be a lot of work. Does there already exist a script which sums all surfaces in a map? Thanks for your help,

Niels

adamw on 12/4/2006 6:50 AM (#31445)

Try this (requires Surface Tools because of the call to Surface.TransformWith):

```
'VBScript
Sub Main
  Set sum = Nothing
  For Each c In Document.ComponentSet
    If c.Type = ComponentSurface Then
      Set sum = Document.NewSurface("Sum", c.Width, c.Height, _
        ValueTypeFloat64, c.CoordinateSystem)
      Exit For
    End If
  Next
  If sum Is Nothing Then
    Exit Sub
  End If
  For Each c In Document.ComponentSet
    If c.Type = ComponentSurface And c.ID <> sum.ID Then
      sum.PixelSet.TransformWith "[" & sum.Name & "] + [" & c.Name & "]"
    End If
  Next
End Sub
```

Extract the Surface Height for Each Point in Drawing

http://69.17.46.171/Site/Thread.aspx?id=1196&ti=632188251000000000

adamw on 4/28/2004 7:51 AM (#2447)

The following script reports a height for each point in a drawing "D" on a surface "S":

```
Sub Main
  Set cset = Application.ActiveDocument.ComponentSet
  Set drw = cset("D")
  Set srf = cset("S")

    ' prepare to convert coordinates from surface to drawing
  Set cnv = Application.NewCoordinateConverter
  cnv.Prepare drw, srf
```

```
    ' find out height for each point
  For Each obj In drw.ObjectSet
    If obj.Type = ObjectPoint Then
      Set pt = obj.Geom.BranchSet(0).PointSet(0)
      cnv.Convert pt

        ' WARNING: check if pt is out of surface
      Set px = srf.PixelSet(srf.PixelSet.ItemByXY(pt.X, srf.Height - pt.Y - 1))

      Application.MessageBox obj.ID & " : " & px.Value

    End If

  Next

End Sub
```

Hung on 4/28/2004 11:41 AM (#2458)

Thanks, Adam! That seems to work although some values were off a bit as compared to the values from Transfer Height function. They are very close. :-)

So when you convert the drawing coordinate system to the same one as in the surface, do the point coordinates (pt.X, pt.Y) in the drawing equate to the pixel "X Offset (I)" and "Y Offset (I)" of the surface?

I'm trying to understand the intrinsic fields of the surface.

adamw on 4/29/2004 8:45 AM (#2463)

Hung - 2004-04-28 8:41 AM - So when you convert the drawing coordinate system to the same one as in the surface, do the point coordinates (pt.X, pt.Y) in the drawing equate to the pixel "X Offset (I)" and "Y Offset (I)" of the surface?

It is slightly more complicated than that. The coordinate system of the surface is X (I) and Y (I), however, the ItemByXY method of the PixelSet object accepts the equivalents of X Offset (I) and Y Offset (I).

Calculating Hillshade from a DEM

http://forum.manifold.net/Site/Thread.aspx?id=31964&ti=633016799144170000

chrismarx on 12/12/2006 12:12 PM (#31965)

OK, another obscure raster calculation! Anybody out there know if Manifold can calculate hillshades from a DEM? The word "hillshade" is not even in the help file, so I'm looking for any help finding the right formula.

adamw on 12/14/2006 7:58 AM (#32178)

Here is one method:

```
//C++
sunHorizontal = Deg2Rad(0);
sunVertical = Deg2Rad(45); // say, 45 degrees up Earth

shade = cos(sunVertical);
slope = ComputeSlope(surface, x, y); // returns radians

If (slope != 0)
  {
    aspect = ComputeAspect(surface, x, y); // returns radians
    shade = cos(sunVertical)*cos(slope) +
      sin(sunVertical) * sin(slope) * cos(sunHorizontal-aspect);
    if (shade < 0) shade = 0;
  }

// applying shade:
// intensity = (z - minZ) * (maxZ - minZ) * shade; // 0..1
```

Nick Verge on 12/14/2006 10:42 AM (#32208)

Is not hillshading (let's call it shadowing) just the the inverse of the amount of insolation a slope will recieve for a given solar incidence radince angle and azimouth?

See the morphometric variables thread - On the website of Peter Sharry, linked to therein, one of the morphometric variables described is surface insolation (a contraction of "incoming solar radiation")

http://www.giseco.info/pb/wp_d4052550/wp_d4052550.html?0.5172811002358266

Import / Export

Import Surfaces, Set Surface Projections, then Merge to One Surface

IMPORTANT NOTE FROM KlausDE on GeoReference May 18, 2006:
6.5->7.0: new CoordinateParameterSet
KlausDE at 5/18/2006 5:33 AM (#21328)
With Manifold 7.0 the structure of the CoordinateSystemParameterSet has changed.
The Index of Parameters is shifted and that results in mdsumner's nice script to "Set Current Projection" to fail. We have to adjust the script to:

```
localScaleX = params(9)
localScaleY = params(11)
localOffsetX = params(13)
localOffsetY = params(15)
```

(reading and writing a few lines later)

or explicitly change to

```
localScaleX = params(params.ItemByName("localScaleX")) ...
```

FOLLOWUP NOTE FROM ADAM
adamw at 5/18/2006 9:02 PM (#21357)
I suggest using the second variation, possibly simplified to:

```
'VBScript
localScaleX = params("localScaleX")
```

Names are much more robust than indices. There is still no guarantee that the names will not change between versions, but they are much less likely to change than indices.

Surfaces - Import, Projection, Merging, scripting
12/09/2003 10:13 PM
http://www.georeference.org/Forums/tabid/71/forumid/7/postid/477/view/topic/Default.aspx

Hello, there's been various discussions regarding automation and performance with surfaces in Manifold lately, so I just thought I'd relate a happy story regarding something just done on a

Dell P4, 2.60GHz with 1.0 Gb RAM
Windows XP Pro 2002, SP1
Manifold 5.50 Pro SP2

The scripts used are provided and discussed below

I've indicated very rough guesses at the time taken for each step, the total time to (re)write the scripts was less than an hour (after 18 mths experience!). Most of the scripting is basic modifications on the Manifold examples. The.map files of ~500Mb were saved a few times each of which took <10 min

There were 20 DEM surfaces from .ADF format

- (Total ~600Mb)
- these came from some obscure ESRI source, most had 1m pixels, some had 2m pixels
- each had varying amounts of invisible pixels

These were imported all to Manifold by script

- (Total .map ~450Mb
- this took about 20min

Current projection of all was incorrect, but localScale X/Y and localOffset X/Y were correct (so only projection needed setting in "Current Projection")

The current projection of all were set to the correct UTM 49 (S) by script

- localOffset and localScale values were preserved as variables in the script before changing Current projection
- this took a few seconds

A surface ("Merged") was created to cover extent of all separate surfaces to 2m pixels (this was done "by hand" in the GUI)

- it was ensured that the local offset X/Y of merged was correct
- all pixels in "Merged" were made invisible since the default zero would conflict with some DEM heights
- surface size was 8750x16681, type was Float32
- this took ~10min for Manifold to create the surface and a few minutes to make sure the offset and scale and extent were OK

All separate surfaces were then copied and pasted to "Merged" using a script

- this of course resampled to 2m pixels where necessary
- each surface in turn was deleted after copying to save memory(here must iterate from ComponentSet.Count down since this changes on deletions
- I always make this mistake at first!)
- this took ~20min

If we'd done the same for a surface at 1m pixels the size would be >2Gb but I get the impression that it would have succeeded in doing so in ~4X the time. Next time I'll try to remember to have the scripts record the time taken! And to set ActiveDocument.BatchUpdates to TRUE . . .

JScript to import the separate surfaces

Note that I have a particular filter for the file names (which are all the same! but are kept in separate folders), it only imports if the file name is "w001001.adf". Here you will need your own filter for particular files. (The adf importer will pull in any .adf file as the entire surface for the ones I have).

This script will delve into the specified directory and all subdirectories to import files that fit the criteria . . ..

```
{code jscript - temp tag to be replaced}
//'************************************************************
//
//' Author: Michael D. Sumner Date: December 10 2003 Version: 1.0.1
//' mikemiq@yahoo.com.au
//' Platform: Manifold 5.5 SP2. Not tested on any other.
//'
//' Language: JavaScript (Manifold System)
//'
//' Permission to use, modify and distribute this code is hereby granted, providing
this
//' entire comment section is included. No warranty is expressed or implied.

//'************************************************************

//' Ask user for starting folder and import it
function Main() {

caption = "Batch Import";

//' query name of folder to import
folder = Application.InputBox("Enter folder:", caption);
if (!folder.length)
return;

//' create file system object and obtain folder
fso = new ActiveXObject("Scripting.FileSystemObject");
fld = fso.GetFolder(folder);
if (fld == null) {
Application.MessageBox("Folder does not exist.", caption);
return;
}

Import(fld);
}

' ************************************************************************

//' Import folder
function Import(folder) {

var imp = Application.NewImport("ADF");


//' examine files with "adf" extension - this loop is a bit overkill, I just haven't
//' changed it from the example except to make it work for me
var files = new Enumerator(folder.Files);
```

```
for (; !files.atEnd(); files.moveNext()) {
file = files.item();
name = file.Name;
dotPos = name.lastIndexOf(".");
if (dotPos >= 0) {
//' I only import "w001001.adf" ext = name.substring(dotPos+1,
name.length).toLowerCase();
if (name == "w001001.adf")
imp.Import(file, PromptNone);
}
}

//' import subfolders
var folders = new Enumerator(folder.SubFolders);
for (; !folders.atEnd(); folders.moveNext())
Import(folders.item());
}
{/code - temp tag to be replaced}
```

VBScript to set the current projection of all surfaces

Note that this assumes that projection and datum of all are the same, and that the scale and offset values are correctly specified already - this sometimes happens with certain formats that save some but not all projection metadata. (I've had this with TIF before).

```
{code - temp tag to be replaced}
'*************************************************************

' Author: Michael D. Sumner Date: December 10 2003 Version: 1.0.1
' mikemiq@yahoo.com.au
' Platform: Manifold 5.5 SP2. Not tested on any other.
'
' Language: VBScript (Manifold System)
'
' Permission to use, modify and distribute this code is hereby granted,
' providing this entire comment section is included. No warranty is
' expressed or implied.

'*************************************************************

Sub Main
'define the projection and datum that I know my data are in
GridToLoad = "Universal Transverse Mercator - Zone 49 (S)"
DatumToLoad = "World Geodetic 1984 (WGS84)"

'iterate over all components, set the Current projection of each surface
' - saving the offset and scale parameters which are good

Set comps = Application.ActiveDocument.ComponentSet
for each comp in comps
   If comp.Type = ComponentSurface Then
         ' retrieve the CS and parameters of the surface
      Set loader = comp.CoordinateSystem
      Set params = loader.Parameters

         ' NOTE that these parameters may be referenced by name using
         ' params(param.ItemByName("localScaleX"))
```

```
          ' but here I just have the literal index

          ' SAVE the offset and scale from the imported projection
          ' ORIGINAL VERSION BELOW
          '      localScaleX = params(6).Value
          '      localScaleY = params(8).Value
          '      localOffsetX = params(10).Value
          '      localOffsetY = params(12).Value

         ' NEW CODE BASED UPON V7 CHANGES - SEE NOTE AT BEGINNING
         ' OF THIS SECTION FROM KlausDE
         ' YOU CAN ALSO EXPLICITLY CHANGE THE PARAMETRES:
         ' localScaleX = params(params.ItemByName("localScaleX"))
         '
          '       localScaleX = params(9)
          '       localScaleY = params(11)
          '       localOffsetX = params(13)
          '       localOffsetY = params(15)

         ' BEST APPRAOCH SUGGESTED BY KlausDE and Adam
            ' localScaleX = params(params.ItemByName("localScaleX"))
            '.OR the shorter variation provided by Adam.
        localScaleX = params("localScaleX")
        localScaleY = params("localScaleY")
        localOffsetX = params("localOffsetX")
        localOffsetY = params("localOffsetY")



          ' change the current projection of the surface to the desired one
          ' and respecify the scale and offset from the import

      loader.Load(GridToLoad)
      loader.Datum.Load(DatumToLoad)

          ' Pre-V7 settings
          '      params(6).Value = localScaleX
          '      params(8).Value = localScaleY
          '      params(10).Value = localOffsetX
          '      params(12).Value = localOffsetY

          'V7
          '      params(9).Value = localScaleX
          '      params(11).Value = localScaleY
          '      params(13).Value = localOffsetX
          '      params(15).Value = localOffsetY

      params("localScaleX")  = localScaleX
      params("localScaleY")  = localScaleY
      params("localOffsetX") = localOffsetX
      params("localOffsetY") = localOffsetY


   End If
Next

End Sub
```

```
{/code - temp tag to be replaced}
```

VBScript to merge all surfaces to one
Note that this assumes you have created a single surface "Merged" that covers the extent of all surfaces and is at least close in pixel size. **However**, it doesn't have to have this constraint - the paste will resize "Merged" and resample the pixels of the copied surface appropriately - (as far as the "Confirm expanding images or surfaces when pasting" option in Tools - Options - Confirmations is turned on. - thanks AdamW)

- but there is only so much it can do, it will be impossible to resize above a certain size and performance is much better if resampling and resizing is not necessary.

```
{code - temp tag to be replaced}
'*************************************************************

' Author: Michael D. Sumner Date: December 10 2003 Version: 1.0.1
' mikemiq@yahoo.com.au
' Platform: Manifold 5.5 SP2. Not tested on any other.
'
' Language: VBScript (Manifold System)
'
' Permission to use, modify and distribute this code is hereby granted, providing this
' entire comment section is included. No warranty is expressed or implied.

'*************************************************************
Sub Main
Set comps = Application.ActiveDocument.ComponentSet

'set the merged surface
Set merged = comps("Merged")

'iterate over each surface in component, copying contents to merged
For each cmp in comps

If cmp.Name <> merged.Name And cmp.Type = ComponentSurface Then

cmp.Copy FALSE 'copy entire component (not just selection)
merged.Paste FALSE 'paste but don't replace selection
comps.Remove comps.ItemByName(cmp.Name)

End If

Next
End Sub

{/code - temp tag to be replaced}
```

12/10/2003 1:30 AM Alert
An addition:

mdsumner - 2003-12-09 7:13 PM - However , it doesn't have to have this constraint - the paste will resize "Merged" and resample the pixels of the copied surface appropriately ...

As far as the "Confirm expanding images or surfaces when pasting" option in Tools - Options - Confirmations is turned on.

Adam Wachowski
Manifold Development Team

06/05/2005 9:35 PM Alert
Hello,

The "VBScript to set the current projection of all surfaces" works great for my needs--except that it changes the units from "Arc-second" to "degrees." How would I edit the script so that arc-seconds is retained?

Thanks in advance

Justin

06/10/2005 8:32 AM Alert
There is a Unit object that is part of the CoordinateSystem object. Try experimenting with that.

If writing small, experimental scripts is causing you difficulty, then you may need to rephrase your query.

06/10/2005 9:01 AM Alert
Hi Mike,

Yes, I am pretty-well clueless about programming with Visual Basic, or any other language for that matter.

"There is a Unit object that is part of the CoordinateSystem object. Try experimenting with that."

Problem is that I have no idea >how< to experiment with it. I suppose when I have more time I could--and will sometime in the future--spend two or three weeks or more reading and studying to gain a rudimentary knowledge of all the concepts, syntax, etc., of VB programming, but for now was sort of looking for some ready-made code that I could cut and paste into the script. I was hoping that someone who already has all that programming knowledge and expertise could whip out the code in a few minutes and post it.

I can usually take a generic script and figure out how to customize it a little for my particular needs, but adding specific code like this is something currently beyond my comprehension.

Justin

Import External Images Using Drawing Index

http://forum.manifold.net/Site/Thread.aspx?id=1967&ti=633005081137170000

mdsumner on 11/29/2006 2:29 PM (#1968)

Hello,
here is a script (with .map data for example purposes) for using scripts to import external image files based on a drawing index.

It's in response to this thread on Manifold-L

http://lists.directionsmag.com/discussion/read.php?f=29&i=35753&loc=0&t=35753

You need to have some external images for it to work, I have supplied some with instruction in the .map file.

Hope it's useful,

cheers, Mike.

```
' ImportSelectedTile
'
' Author: Michael D. Sumner Date:March 24 2004
' Language: VBScript Version: 1.0.1
' Platform: Manifold 5.5 SP2. Not tested in any other.
'
' Permission to use, modify and distribute this code is hereby granted, providing this
' entire comment section is included. No warranty is expressed or implied.
'
' Usage:
' (Assumes the existence of an external set of images as specified in "Tiles
Table"["FileName"],
' export images in "Image Folder" to file names in table.)
' Open drawing "Tiles" or "Map" with "Tiles" as active window.
' Select one of the tiles, run script. The image should be imported and properly
georeferenced.
'
' Notes:
' This should work for any image format and any projection, but I've not tested it.
' I've successfully used a similar process for images in UTM, but that had a lot of
other
' stuff implemented on top for a separate image serving application.
' The setting of projection stuff is redundant if you have reliably georeferenced
images in
' a good format that retains projection metadata.
'*************************************************************************
```

```
Sub Main
  'get the active window, drawing or drawing in map
  Set Window = Application.WindowSet.ActiveWindow
  Set Component = Window.ActiveComponent

  If Component.Type <> ComponentDrawing Then
    Application.Messagebox "Active window must be a drawing"
    Exit sub
  End If

  'get the selection in the drawing
  Set selectImg = Component.Selection

  If selectImg.Count = 0 Then
    Application.Messagebox "No image tiles selected, exiting"
    Exit Sub
  End If

  ' just import the first one
  Set imgArea = selectImg(0)
  Set imgRecord =
Component.OwnedTable.RecordSet(Component.OwnedTable.RecordSet.ItemByID(imgArea.ID))
  FileName = imgRecord.Data("FileName")
  Ext = UCase(Right(FileName,3))

  If Ext = "TIF" Then Ext = "TIFF"

  Set Importer = Application.NewImport(Ext)
  Importer.Import FileName

  Set Image = Importer.ComponentSet(0)

  ' set the image's CURRENT projection
  ProjToLoad = imgRecord.Data("Projection")
  'bug in SP2 requires use of first branch's Rect object, rather than that of geom
  sclX = imgArea.Geom.BranchSet(0).Box.Width / Image.Width
  sclY = imgArea.Geom.BranchSet(0).Box.Height / Image.Height
  offX = imgArea.Geom.BranchSet(0).Box.Xmin
  offY = imgArea.Geom.BranchSet(0).Box.Ymin

  Set Proj = Application.NewCoordinateSystem(ProjToLoad)
  Proj.Parameters(Proj.Parameters.ItemByName("localScaleX")).value = sclX
  Proj.Parameters(Proj.Parameters.ItemByName("localScaleY")).value = sclY
  Proj.Parameters(Proj.Parameters.ItemByName("localOffsetX")).value = offX
  Proj.Parameters(Proj.Parameters.ItemByName("localOffsetY")).value = offY

  Image.CoordinateSystem = Proj
  Image.Open
End Sub
```

Export (to Manifold Projects) Internal Drawing Tiles Using Drawing Index

http://forum.manifold.net/Site/Thread.aspx?id=1967&ti=633005081137170000

mdsumner on 11/29/2006 2:28 PM (#16166)

Here I do something similar for export (to separate Manifold projects, but should be easily adaptable for images):

WARNING: this is unsupported code, it worked once upon a time for me but I've not used it in a while.

```
Sub Main

  dir = "C:\Documents and Settings\mdsumner\My Documents\CODE\Manifold\ImageTiles\"

  Set doc = Application.ActiveDocument
  Set comps = doc.ComponentSet
  Set dwg2Render = comps("CountriesAreas")
  Set dwgTiles = comps("Tiles")


  '' this section will render an image
  width  = 400
  height = 400

  for each tile in dwgTiles.ObjectSet
    'Set tile = dwgTiles.ObjectSet(0)
    ' here get the tile number from the recordset
    'imgName = "Tile1"
    Set record = _
dwgTiles.OwnedTable.RecordSet.Item(dwgTiles.OwnedTable.RecordSet.ItemByID(tile.ID))
    imgName = "Tile_" & record.Data("tilenumber")

    '' need to use BranchSet here, bug in 5.50 SP2
    Set tileBox = tile.Geom.BranchSet(0).Box
    dwg2Render.RenderAreaTo imgName,width,height,tileBox,TRUE

    Set imgTile = comps(imgName)
  '' this section will create a new project file with
  '' an image, then remove it from this one
    docFile = dir & imgName & ".map"

    record.Data("projectName") = docFile
    Set newDoc = Application.NewDocument(docFile,FALSE)

    imgTile.Copy
    newDoc.Paste
    newDoc.Save()

    comps.Remove comps.ItemByName(imgName)

  Next

End Sub
```

Export a Surface Straight to Image Format

Manifold-L – Jan 18, 2006
Author: Michael Sumner
Date: 01-18-2006 18:48
http://lists.directionsmag.com/discussion/read.php?f=29&i=41362&t=41355

I thought you would need to render the surfaces first as an image, then export those to JPEG - but not apparently, you can export a surface straight to image format. Note that you can't do this in the GUI . . .

There are examples on www.georeference.org in the archived site that will help you obtain all the surfaces in a folder, and export each to (image) file. Post further questions there if you need help. Here's the basic (VB)script for doing one explicitly:

```
Sub Main
    Set doc = Application.ActiveDocument
    Set comps = doc.ComponentSet
    Set comp = comps("Surface")

    Set expt = Application.NewExport("JPEG")
    expt.Export comp, "C:/temp/image.jpg",
End Sub
```

Exporting Surfaces as Images

http://69.17.46.171/Site/Thread.aspx?id=18134&ti=632736022437170000
nielsraes on 1/23/2006 8:24 AM (#18135)
KlausDE on 1/23/2006 8:37 AM (#18136)

```
Sub Main
  Set doc = Application.ActiveDocument
  Set comps = doc.ComponentSet
  Set comp = comps("Surface")

  Set expt = Application.NewExport("JPEG")
  expt.Export comp, "C:/temp/image.jpg", FALSE
End Sub
```

Exporting All Images in a Project

Manifold©-L
Author: Adam Wachowski
Date: 03-04-2006 09:27
http://lists.directionsmag.com/discussion/read.php?f=29&i=41743&t=41727

```
Sub Main
  Set exporter = Application.NewExport("ECW")
  Set components = Application.ActiveDocument.ComponentSet
  For i = 0 to components.Count-1
    Set theComponent = components(i)
    If theComponent.typeName = "Image" Then
      exporter.CompressionRatio = 10
      exporter.Export theComponent, theComponent.Name + ".ecw", PromptNone
    End If
  Next
End Sub
```

Exporting Geometry as SHP MetaData

From: James Burn
Sent: Thursday, March 02, 2006 9:49 PM
To: zjmcypan@pty.com; Manifold List
Subject: Re: [Manifold-l] [Spam] Metadata
http://lists.directionsmag.com/discussion/read.php?f=29&i=41732&t=41726

You can export the geometry as shp files; do a search on the forum and you'll find a metadata script , and you just need to export the proper XML file for the metadata, right? If you search the Manifold forum, you'll find a script from Bob Heitzman for creating metadata.

Cheers.

James Burn BSc, GIS(pg), AScT

```
Option Explicit
' MetaReport20021013
' Author: Bob Heitzman, CopyLeft 2002.10
' Please report any problems to rheitzman@hotmail.com
' Produces and XML file that includes the mandatory FGDC Metadata

' Most of the static fields have been set to "ToDo". You can edit
' those in the script below to insert your static information or
' use a text editor afterwards to update the information.

' I included the <place> elements as they are very common although not mandatory.

' I inserted the bulk of the projection information in the <abstract>
' element as it is not mandatory but is very useful. The FGDC encoding
```

```
' of the projection information is too complex to generate for fun.

' Several portions of the script are commented out and expect you
' to indicate what you want. For example in Man() there is an
' If statement that allows you to select to produce a report for
' all components if set to True or just the open component if False.

Dim manDoc
Const LatLongDigits = 4

Sub Main()
    Dim window, windows
    Dim manComponents
    Dim manComp
    Dim Index, s

    set manDoc = Application.ActiveDocument

    ' If True report on all components
            If False then
                        Dim dtStart
                        dtStart = Now()
            Set manComponents = manDoc.ComponentSet

            For Index = 0 To manComponents.Count-1
                        Set manComp = manComponents(Index)
                        Application.StatusText = manComp.TypeName & ":" & manComp.Name
& " " & Index+1 & " of " & manComponents.Count
                        ' Application.Messagebox manComp.TypeName & ":" & manComp.Name
                        s = MetadataMandatoryRpt(manComp)
            Next
                Application.Messagebox "Done with batch:" &
DateDiff("s",dtStart,Now()) & " seconds"
            Else
    ' report on active component
                set windows = Application.WindowSet

                ' -- fail if there is no opened windows
                If windows.Count = 0 Then
                    Application.Messagebox ("No active component")
                    Exit Sub
                End If

                set window = windows.ActiveWindow
                set manComp = window.Component

                        Application.StatusText = manComp.TypeName & ":" & manComp.Name

                s = MetadataMandatoryRpt(manComp)
                Application.Messagebox ("Created Report: " & s)

            End If

            Application.StatusText = ""

End Sub
```

```
' ************************************************************************

Function MetadataMandatoryRpt(ByVal manComp)
    Dim Report
    Dim manObjSet
    Dim geom
            Dim pixelSet
    Dim rect
    Dim params
    Dim s, sRptName
    Dim ptMax' As Manifold.Point
    Dim ptMin 'As Manifold.Point

    Set ptMax = Application.NewPoint
    Set ptMin = Application.NewPoint

    Select Case manComp.TypeName
    Case "Drawing", "Surface", "Image" ' ignore others
        ' -- create metadata report
        s = manComp.Name & " Metadata"
        set Report = manDoc.NewComments(s)
        sRptName = Report.Name

                        Application.StatusText = Application.StatusText & "."

        Report.AddText("<?xml version=""1.0""?>" & vbCrLf)

        'XSLT References - pick one to apply a format style to your XML file
                        ' the ESRI refernces only partially work as they require
either very
                        ' specfic FGDC tags or ESRI extensions.
                        ' THe FGDC STandard Format is recommened

        'FGDC Standard Format - public domain
        Report.AddText("<?xml-stylesheet type=""text/xsl""
href=""http://www.fgdc.gov/FGDC_V2.xsl""?>" & vbCrLf)
        'ESRI Tabbed Dialog
        '    Report.AddText "<?xml-stylesheet type=""text/xsl""
href=""http://www.fgdc.gov/metadata/metaxml/fgdc_esri.xsl""?>"
        'ESRI 'FAQ Format
        '    Report.AddText "<?xml-stylesheet type=""text/xsl""
href=""http://www.fgdc.gov/metadata/metaxml/fgdc_faq.xsl""?>"
        'ERIS Condensed - Geography Netork
        '    Report.AddText "<?xml-stylesheet type=""text/xsl""
href=""http://www.fgdc.gov/metadata/metaxml/fgdc_geography_network.xsl""?>"

        Report.AddText("<metadata>" & vbCrLf)
        Report.AddText("<idinfo>" & vbCrLf)
        Report.AddText("<citation>" & vbCrLf)
        Report.AddText("<citeinfo>" & vbCrLf)
        Report.AddText("<origin>ToDo</origin>" & vbCrLf)
        Report.AddText("<pubdate>" & YYYYMMDD_Date(Now()) & "</pubdate>" & vbCrLf)
        Report.AddText("<title>" & manComp.Name & "</title>" & vbCrLf)
        Report.AddText("</citeinfo>" & vbCrLf)
        Report.AddText("</citation>" & vbCrLf)
        Report.AddText("<descript>" & vbCrLf)
```

```
                        Report.AddText("<abstract>Component Description Property: " &
manComp.Description & "</abstract>" & vbCrLf)

                        Application.StatusText = Application.StatusText & "."
        s = ComponentReport(manComp)
                        Report.AddText("<supplinf>" & s & "</supplinf>" & vbCrLf)
                        Application.StatusText = Application.StatusText & "."

        Report.AddText("<purpose>ToDo</purpose>" & vbCrLf)
        Report.AddText("</descript>" & vbCrLf)
        Report.AddText("<timeperd>" & vbCrLf)
        Report.AddText("<timeinfo>" & vbCrLf)
        Report.AddText("<sngdate>" & vbCrLf)
        Report.AddText("<caldate>" & YYYYMMDD_Date(Now()) & "</caldate>" & vbCrLf)
        Report.AddText("<time>" & HHMM_Time(Now()) & "</time>" & vbCrLf)
        Report.AddText("</sngdate>" & vbCrLf)
        Report.AddText("</timeinfo>" & vbCrLf)
        Report.AddText("<current>ToDo ground condition | publication date | free
text</current>" & vbCrLf)
        Report.AddText("</timeperd>" & vbCrLf)
        Report.AddText("<status>" & vbCrLf)
' pick one        Report.AddText("<progress>ToDo Complete | In work |
Planned</progress>" & vbCrLf)
        Report.AddText("<progress>In work</progress>" & vbCrLf)
        Report.AddText("<update>ToDo Continually | Daily | Weekly | Monthly | Annually
| Unknown | As needed | Irregular | None planned | free text</update>" & vbCrLf)
        Report.AddText("</status>" & vbCrLf)

        Report.AddText("<spdom>" & vbCrLf)
        Report.AddText("<bounding>" & vbCrLf)
                        Application.StatusText = Application.StatusText & "."
        ' -- append drawing to report
'            If manComp.TypeName = "Drawing" Then
        set params = manComp.CoordinateSystem.Parameters
                        Set rect = Nothing

                        Select Case manComp.TypeName
                        Case "Drawing"
            set manObjSet = manComp.ObjectSet
            set geom = manObjSet.GeomSet
            set rect = geom.EnclosingRect()
                        Case "Image", "Surface"
                                    dim obj
                                    Set obj = manComp
                                    set rect = obj.PixelSet '.Box
        Case Else ' not done yet
            Report.AddText("<westbc>ToDo</westbc>" & vbCrLf)
            Report.AddText("<eastbc>ToDo</eastbc>" & vbCrLf)
            Report.AddText("<northbc>ToDo</northbc>" & vbCrLf)
            Report.AddText("<southbc>ToDo</southbc>" & vbCrLf)
        End Select
                        If Not (rect Is Nothing) then
'                                    If manComp.CoordinateSystem.Name = "Latitude /
Longitude" Then
```

```
'                        Report.AddText("<westbc>" & FormatNumber(rect.Box.XMin *
params(params.ItemByName("localScaleX")).value, LatLongDigits, 0, 0, 0) & "</westbc>"
& vbCrLf)
'                Report.AddText("<eastbc>" & FormatNumber(rect.Box.XMax *
params(params.ItemByName("localScaleX")).value, LatLongDigits, 0, 0, 0) & "</eastbc>"
& vbCrLf)
'                Report.AddText("<northbc>" & FormatNumber(rect.Box.YMax *
params(params.ItemByName("localScaleY")).value, LatLongDigits, 0, 0, 0) & "</northbc>"
& vbCrLf)
'                Report.AddText("<southbc>" & FormatNumber(rect.Box.YMin *
params(params.ItemByName("localScaleY")).value, LatLongDigits, 0, 0, 0) & "</southbc>"
& vbCrLf)
'                                    Else
'                Dim ptMax' As Manifold.Point
'                Dim ptMin 'As Manifold.Point
'                Set ptMax = Application.NewPoint
'                Set ptMin = Application.NewPoint
                ptMax.x = rect.Box.XMax
                ptMax.Y = rect.Box.YMax
                Set ptMax = manComp.CoordinateSystem.ConvertInverse(ptMax)
                ptMin.x = rect.Box.XMin
                ptMin.Y = rect.Box.YMin
                Set ptMin = manComp.CoordinateSystem.ConvertInverse(ptMin)
                        Report.AddText("<westbc>" & FormatNumber(ptMin.x,
LatLongDigits, 0, 0, 0) & "</westbc>" & vbCrLf)
                Report.AddText("<eastbc>" & FormatNumber(ptMax.x, LatLongDigits, 0, 0,
0) & "</eastbc>" & vbCrLf)
                Report.AddText("<northbc>" & FormatNumber(ptMax.Y, LatLongDigits, 0,
0, 0) & "</northbc>" & vbCrLf)
                Report.AddText("<southbc>" & FormatNumber(ptMin.Y, LatLongDigits, 0,
0, 0) & "</southbc>" & vbCrLf)
 '                                   End If
                        end if
                        Application.StatusText = Application.StatusText & "."
        Report.AddText("</bounding>" & vbCrLf)
        Report.AddText("</spdom>" & vbCrLf)

        Report.AddText("<keywords>" & vbCrLf)
        Report.AddText("<theme>" & vbCrLf)
        Report.AddText("<themekt>ToDo: None | free text</themekt>" & vbCrLf)
        Report.AddText("<themekey>ToDo: keyword, keyword, etc.</themekey>" & vbCrLf)
        Report.AddText("</theme>" & vbCrLf)

                        ' <Place> is not mandatory but common
        Report.AddText("<place>" & vbCrLf)
        Report.AddText("<placekt>ToDo: None | Geographic Names Information System |
free text</placekt>" & vbCrLf)
        Report.AddText("<placekey>ToDo</placekey>" & vbCrLf)
        Report.AddText("</place>" & vbCrLf)

        Report.AddText("</keywords>" & vbCrLf)
        Report.AddText("<accconst>ToDo</accconst>" & vbCrLf)
        Report.AddText("<useconst>ToDo</useconst>" & vbCrLf)
        Report.AddText("</idinfo>" & vbCrLf)
        Report.AddText("<metainfo>" & vbCrLf)
        Report.AddText("<metd>" & YYYYMMDD_Date(Now()) & "</metd>" & vbCrLf)
```

```
' Contact Info - insert you contact info in place of ToDo
        Report.AddText("<metc>" & vbCrLf)
        Report.AddText("<cntinfo>" & vbCrLf)

' choose one of the next two sections
' unless you do mp will return the error:
'    Error (line xx): Contact_Information permits only one of Contact_Person_Primary
' or Contact_Organization_Primary
                        ' Contact_Person_Primary
        Report.AddText("<cntperp>" & vbCrLf)
        Report.AddText("<cntper>ToDo</cntper>" & vbCrLf)
        Report.AddText("<cntorg>ToDo?</cntorg>" & vbCrLf)
        Report.AddText("</cntperp>" & vbCrLf)
                        ' Contact_Organization_Primary
'        Report.AddText("<cntorgp>" & vbCrLf)
'        Report.AddText("<cntorg>ToDo</cntorg>" & vbCrLf)
'        Report.AddText("<cntper>ToDo?</cntper>" & vbCrLf)
'        Report.AddText("</cntorgp>" & vbCrLf)

        Report.AddText("<cntaddr>" & vbCrLf)
                        ' choose and address type or make one up
        Report.AddText("<addrtype>ToDo: mailing | physical | mailing and physical |
free text</addrtype>" & vbCrLf)
                        ' <address> may be repeated if required
        Report.AddText("<address>ToDo*</address>" & vbCrLf)
        Report.AddText("<city>ToDo</city>" & vbCrLf)
        Report.AddText("<state>ToDo</state>" & vbCrLf)
        Report.AddText("<postal>ToDo</postal>" & vbCrLf)
        Report.AddText("<country>ToDo?</country>" & vbCrLf) ' optional
        Report.AddText("</cntaddr>" & vbCrLf)
        Report.AddText("<cntvoice>ToDo</cntvoice>" & vbCrLf)
        Report.AddText("</cntinfo>" & vbCrLf)
        Report.AddText("</metc>" & vbCrLf)

        Report.AddText("<metstdn>FGDC Content Standards for Digital Geospatial
Metadata</metstdn>" & vbCrLf)
        Report.AddText("<metstdv>FGDC-STD-001-1998</metstdv>" & vbCrLf)
        Report.AddText("</metainfo>" & vbCrLf)
        Report.AddText("</metadata>" & vbCrLf)

    Case Else
        sRptName = "Active window should be a geographic component."
    End Select
            Application.StatusText = Application.StatusText & "."

            ' If True write output to a file,
            ' add/remove comments to create one test.xml file or a batch of files
            ' also edit the pathname to match your PC
            If False Then
                        Select Case manComp.TypeName
            Case "Drawing", "Surface", "Image" ' ignore others

                                    ' this code creates a test.xml file
                                    ' This code should be modified to point to your
folder/file preferences
                                    Dim FSO, Stream
```

```
                                    Set FSO =
CreateObject("Scripting.FileSystemObject")
                                    ' this code creates a 'test.xml' file useful for
testing
                                    Set Stream =
FSO.CreateTextFile("D:\GIS_DATA\Metadata\test\test.xml" , True)
                                    Stream.Write Report.Text
                                    Stream.Close

                                    ' This code creates a file using the Report.Name
                                    '  Note that the Report.Name may not be a valid
fle name
                                    '  in many instances so watch out for that.
                          '          Set Stream =
FSO.CreateTextFile("D:\GIS_DATA\Metadata\test\" & Report.Name & ".xml", True)
                          '          Stream.Write Report.Text
                          '          Stream.Close
                          End Select
            End If
            MetadataMandatoryRpt = sRptName
End Function

' ************************************************************************

Function ComponentReport(ByVal Component)
    Dim s, i
    Dim System
    Dim params
    Dim Objects
    Dim geom
    Dim rect

    s = Component.TypeName & ":" & Component.Name & vbCrLf
    s = s & "  Note:" & Component.Note & vbCrLf
    s = s & "  Description:" & Component.Description & vbCrLf

    Select Case Component.TypeName
        Case "Drawing", "Surface", "Map", "Image" ' ignore

            Select Case Component.TypeName
                Case "Map", "Layout"
                Case Else
                    s = s & "  Zoom:" & Component.zoommin & "/" & Component.zoommax &
vbCrLf
            End Select

            set System = Component.CoordinateSystem
            set params = Component.CoordinateSystem.Parameters

            ' -- append drawing to report
            If Component.TypeName = "Drawing" Then
                set Objects = Component.ObjectSet
                s = s & "  number of objects: " & CStr(Objects.Count) & vbCrLf
                set geom = Objects.GeomSet
                set rect = geom.EnclosingRect()
                s = s & "  spdom:bounding: " & vbCrLf
```

```
                s = s & "    WestBC: " & FormatNumber(rect.Box.XMin *
params(params.ItemByName("localScaleX")).value, LatLongDigits, 0, 0, 0) & vbCrLf
                s = s & "    EastBC: " & FormatNumber(rect.Box.XMax *
params(params.ItemByName("localScaleX")).value, LatLongDigits, 0, 0, 0) & vbCrLf
                s = s & "    NorthBC: " & FormatNumber(rect.Box.YMax *
params(params.ItemByName("localScaleY")).value, LatLongDigits, 0, 0, 0) & vbCrLf
                s = s & "    SouthBC: " & FormatNumber(rect.Box.YMin *
params(params.ItemByName("localScaleY")).value, LatLongDigits, 0, 0, 0) & vbCrLf
            End If

            s = s & "  Coordinate System: " & System.Preset & vbCrLf
            s = s & "  Datum: " & System.datum.name & vbCrLf

            For i = 0 To params.count - 1
                With params(i)
                    Select Case .name ' supress select params, print others
                        Case "majorAxis", "eccentricity", "centerX", "centerY",
"centerZ"
                            ' ignore these 'cause I don't have a clue as to what they
do
                        Case "scaleX", "localScaleX", "scaleY", "localScaleY",
"falseEasting", "localOffsetX", "falseNorthing", "localOffsetY"
                            ' these are handled with direct calls latter
                        Case Else ' all un accounted for params display
                            s = s & .name & ": " & vbTab & .value & vbCrLf
                    End Select
                End With
            Next

            ' report standard parameters
            With Component.CoordinateSystem.parameters
                s = s & "  Local Offset X/Y: " &
Component.CoordinateSystem.Parameters(.ItemByName("localOffsetX")).value   '& vbcrlf
                s = s & ", " &
Component.CoordinateSystem.Parameters(.ItemByName("localOffsetY")).value & vbCrLf

                s = s & "  Local Scale X/Y:  " &
Component.CoordinateSystem.Parameters(.ItemByName("localScaleX")).value     '& vbcrlf
                s = s & ", " &
Component.CoordinateSystem.Parameters(.ItemByName("localScaleY")).value & vbCrLf

                s = s & "  Scale X/Y(Ix):    " &
Component.CoordinateSystem.Parameters(.ItemByName("scaleX")).value  '& vbcrlf
                s = s & ", " &
Component.CoordinateSystem.Parameters(.ItemByName("scaleY")).value & vbCrLf

                s = s & "  False Easting/Northing: " &
Component.CoordinateSystem.Parameters(.ItemByName("falseEasting")).value    '& vbcrlf
                s = s & ", " &
Component.CoordinateSystem.Parameters(.ItemByName("falseNorthing")).value & vbCrLf

                ' name, nick name, nick name area
                s = s & "  Units: " & System.unit.name & ", " & System.unit.nickname &
", " & System.unit.nicknamearea & vbCrLf

                ' Lat/lon vs. Linear T/F,  Scale: multiplier to meters or to degrees
```

```
                s = s & "         Lat/Lon(I)=" & System.unit.islatlon & ", Scale to
m/deg(I)=" & System.unit.scale & vbCrLf
            End With

    End Select
    ComponentReport = s
End Function

' ************************************************************************

Function HHMM_Time(ByVal dt)
    Dim s
    If IsDate(dt) Then
        s = Right("0" & Hour(dt), 2)
        s = s & Right("0" & Minute(dt), 2)
    Else
        s = "Not a date"
    End If
    HHMM_Time = s
End Function

Function YYYYMMDD_Date(ByVal dt)
    Dim s
    If IsDate(dt) Then
        s = Year(dt)
        s = s & Right("0" & Month(dt), 2)
        s = s & Right("0" & Day(dt), 2)
    Else
        s = "Not a date"
    End If
    YYYYMMDD_Date = s
End Function
```

Export Surface Pixels Greater than 0 to XYZ Text File (FileSystemObject)

http://69.17.46.171/Site/Thread.aspx?id=20283&ti=632814571483770000

grmapper at 4/7/2006 10:38 AM (#20284)

Does anyone have a script snippet that would allow me to export to .xyz any surface pixels greater than 0? I could do this interactively but I am interested in learning to script selection and manipulation of pixelsets. Thanks.

adamw at 4/10/2006 7:07 AM (#20349)

Something to get you going:

```
'VBScript
Sub Main
  Set surface = Application.ActiveDocument.ComponentSet("Surface")

  ' create file and export each pixel whose height is greater than 0
  Set fs = CreateObject("Scripting.FileSystemObject")
  Set stream = fs.CreateTextFile("C:\Documents\1.xyz", true)
```

```
  For Each pixel In surface.PixelSet
    If pixel.Value > 0 Then
      stream.WriteLine pixel.X & " " & pixel.Y & " " & pixel.Value
    End If
  Next
  stream.Close
End Sub
```

Generic Exporter

http://www.georeference.org/Forums/tabid/71/forumid/7/tpage/1/view/topic/postid/1063/Default.aspx#1063

WillH
02/01/2004 7:28 PM

The script below is a generic exporter. It should cover basic batch exporting needs. It allows the user to export any component type to any file format supported by Manifold. One feature I would like to add is a prompt once feature where the settings from the first prompt would be applied to all subsequent exports in the batch.

```
' ************************************************************
'Generic Batch Exporter
'William Howell
'November 2003
'For Manifold version 5.50 SP2
'
'Permission to use, modify and distribute this code is hereby granted, providing this
'entire comment section is included. No warranty is expressed or implied.
'
'Exports all components of given type (i.e. Drawing), from specified folder
'within Manifold to folder location in given format (i.e. DXF).
'Values are passed to script by a comma delimted string (without spaces; see
'example below) entered in an input box. It can create one subfolder in the
'based on the export path given
'
'export path, Manifold folder, component type, export format (2-4 letters),
'export prompt
'
'c:\temp\newExportFolder\,myFolder,Drawing,DXF,no
'The above would export to c:\temp\newExportFolder\ (note the ending slash),
'looking in the Manifold folder "myFolder" for components of type Drawing (notice
'case), as DXFs (again, notice case) without export prompting.
'
',,drawing,SHP,yes
'This example will export all drawings in the current project as shapefiles
'to the current folder (the last folder that used for import/export prompting
'for shapefile options on each drawing
'
'Note: For a more dynamic approach to component selection, use a form containing
'a list box.

' ************************************************************
```

```
Sub Main
   Set document = Application.ActiveDocument
   Set compSet = document.ComponentSet
   Set fso = CreateObject("Scripting.FileSystemObject")

     ' Get user values via input box
   entry = InputBox("Enter folder save location, folder to export from, "_
   &"component type to export, export type, prompt"&vbcrlf&_
   "ex: c:\temp\,myFolder,Drawing,DXF,no", _
   "Generic Exporter","c:\temp\,myFolder,Drawing,SHP,no")
   'change this last string to change the default text in the prompt box
   'this would be useful for settings that are used often

     ' Check if string is empty or "Cancel" is pressed, exit if so
   If entry = "" then
      Exit Sub
   End If

     ' Break inputBox string by commas into an array
   settings = Split(entry,",",-1,1)

     ' check if folder path exists, if not then create one subfolder below existing
path
   If fso.FolderExists(settings(0)) = False Then
     Set newFolder = fso.CreateFolder(settings(0))
     If Not IsEmpty(newFolder) Then
        fExists = True
     End If
   End If


     ' Use yes/no from inputBox and convert to 1/0 for converter prompt
   Select Case LCase(settings(4))
      Case "yes" prompt = 1
      Case Else prompt = 0
   End Select

   'option for using components in specified folder vs. entire project
   if settings(1) <> "" then
      set compSet = compSet(settings(1)).children
   end if

     ' drawings have to be handled seperately to check if they are empty
   if settings(2) = "Drawing" then
        ' iterate through every in compSet to match the specified type
      for each comp in compSet
         if comp.TypeName = settings(2) then
              ' empty drawings cannot be exported, so must check
            if not comp.isEmpty() then
               Application.StatusText = "Exporting "&comp.name&" to
"&settings(0)&comp.name&"."&settings(3)
                 ' create exporter of given type
               set genExporter = document.NewExport(ucase(settings(3)))
                 ' export the drawing to the specified folder
               call genExporter.Export(comp,
settings(0)&comp.name&"."&settings(3),prompt)
```

```
            end if
         end if
      next

   else

      for each comp in compSet
         if comp.TypeName = settings(2) then
            Application.StatusText = "Exporting "&comp.name&" to
"&settings(0)&comp.name&"."&settings(3)
              ' create exporter of given type
            set genExporter = document.NewExport(ucase(settings(3)))
              ' export the drawing to the specified folder
            call genExporter.Export(comp,
settings(0)&comp.name&"."&settings(3),prompt)
         end if
      next

   end if

End Sub

' ************************************************************
```

Generate and Export Tile Images

http://69.17.46.171/Site/Thread.aspx?id=21174&ti=632834835570100000

thommy at 5/16/2006 7:19 AM (#21175)

Hello

How can I generate a picture at a certain position (in coordinates) with a certain dissolution in a certain zoom from a MAP (Layer are all Drawings)? I found the Tool - Make image, but there I can't generate a picture from a certain point.

The whole problem in more detail: I must describe a solution, how I can with Manifold generate of a certain zoom stage pictures lining up (with given width and height) over a certain range. Does somebody know a possible solution by coincidence?

mdsumner at 5/16/2006 5:16 PM (#21220)

You can create a grid of points representing the centres of each tile, use Snap To Points to centre the view on the point, zoom to the appropriate scale and then use F6 "Centred view at current scale" with the image dimensions appropriately specified for the tile (which you can work out given your tile spacing and scale). Before the Rect object, and matching RenderAreaTo method I used to do a lot of this . . .

This can be scripted - by iterating across the grid (implicitly specified in code, or as a drawing of area tiles), and creating a Rect object to match your tile extents, then using Map.RenderAreaTo, Rect, to create the image tile.

The second example here does something similar (I can provide better examples if you need it, and I think there are better ones on the archived site somewhere).

http://www.georeference.org/Forums/tabid/71/forumid/7/postid/1711/view/topic/Default.aspx

thommy at 5/16/2006 10:15 PM (#21229)

Thanks very much. Sometimes solutions are so banal, but you don't find!
If you can post a better example, would that me help. Specially included with an example by iterating across the grid(as a drawing of area tiles).

mdsumner at 5/17/2006 5:25 PM (#21284)

This one just uses a table, and modifies the Rect object accordingly for each tile, rather than obtaining it from an actual drawing tile. Also, it collects all the tiles in a folder, and THEN exports them. You'd probably want to export the tiles as they are created, and then delete them in turn.

```
VBScript (not tested recently):
Sub Main
'Find the map, its layout, a grid centre

Set Doc = Application.ActiveDocument
'speed me up Scotty
Doc.BatchUpdates = True

Set Comps = Doc.ComponentSet

'Asuming these components exist already
Set Folder = Comps(Comps.ItemByName("Folder"))
Set Map = Comps(Comps.ItemByName("Map"))
' this is a table of tile centre points, and a column "name"
Set Table = Comps(Comps.ItemByName("Grid Table"))

Set Rect = Application.NewRect

Set Records = Table.RecordSet
cnt = Comps.Count -1
For nn = 0 To Records.Count - 1

Set Rec = Records(nn)
dims = 1570
Xx = Rec.Data("X (I)")
Yy = Rec.Data("Y (I)")
Xcentre = Xx + dims/2
Ycentre = Yy + dims /2

Set Centre = Application.NewPoint
Centre.X = Xcentre
Centre.Y = Ycentre
Rect.Center = Centre
Rect.Xmin = Xx
Rect.Ymin = Yy
Rect.Xmax = Xx + dims
```

```
Rect.Ymax = Yy + dims

' I use the name attribute from the table
Map.RenderAreaTo Rec.Data("name"),dims, dims,Rect, True
cnt = cnt + 1
Set Image = Comps(cnt)
Image.Folder = Folder
Next

Set expt = Application.NewExport("JPEG")
For each comp in Folder.Children
expt.Export comp, "C:/temp/img/" & comp.name & ".jpg", PromptNone

Next
' Comps.Remove(Comps.ItemByName("Layout"))
Doc.BatchUpdates = False
End Sub
```

Export Table to MS EXCEL

http://69.17.46.171/Site/Thread.aspx?id=25853&ti=632911653850930000

ColinD at 8/12/2006 10:03 PM (#25854)

If someone can point me in the right direction to using SQL or script to do the following which I currently do with a lot of selecting copy/pasting and exporting.

I have a table in Manifold that has two columns headed CODE and DIST. CODE contains about 30 different text codes and DIST contains floating point numbers. There are from 3 - >3000 DIST entries per code.

I need to export to xls ending up with columns headed with each code name with the DIST values under that heading.

Thanks

adamw at 8/14/2006 7:18 AM (#25884)

How about:

```
'VBScript
Sub Main
  Set xlApp = CreateObject("Excel.Application")
  xlApp.Workbooks.Add
  Set xl = xlApp.Sheets(1)

  ' obtain records
  Set q = Document.NewQuery("Temp")
  q.Text = "SELECT [Code], [Dist] FROM [T] ORDER BY [Code]"
  q.Run

  ' pass data to Excel
```

```
  code = ""
  column = 1
  row = 1
  For Each r in q.Table.RecordSet
    If r.DataText("Code") = code Then
      row = row + 1
    Else
      code = r.DataText("Code")
      column = column + 1
      row = 2
      xl.Cells(1, column).Value = code
      xl.Cells(1, column).Font.Bold = true
    End If
    xl.Cells(row, column).Value = r.DataText("Dist")
  Next

  ' cleanup
  Document.ComponentSet.Remove(q)

  xlApp.Visible = True
End Sub
```

Export to KML

http://69.17.46.171/Site/Thread.aspx?id=27706&ti=632947912085830000

kgf on 9/21/2006 6:32 PM (#27976)

```
Sub Main

   Dim drawing
   Dim KMLText

   Set components = document.ComponentSet
   Set drawing = components(components.ItemByName("Drawing"))
   Set records = drawing.OwnedTable.RecordSet
   Set KMLText = document.newcomments("RSS KML")

      KMLText.AddText "<?xml version="&CHR(34)&"1.0"&CHR(34)&"
encoding="&CHR(34)&"UTF-8"&CHR(34)&"?>" & vbCrLf
      KMLText.AddText "<kml
xmlns="&CHR(34)&"http://earth.google.com/kml/2.1"&CHR(34)&">" & vbCrLf
      KMLText.AddText "<Folder>"  & vbCrLf
      KMLText.AddText "<name>Your Drawing</name>"  & vbCrLf
      KMLText.AddText "<description>Your Description</description>"  & vbCrLf

   For recordIndex = 0 To records.Count-1

      Application.StatusText = "Scanning table (" & CStr(recordIndex+1) & " of " &
CStr(records.Count) & ")"

         Set record = records(recordIndex)
         KMLText.AddText "<Placemark>"  & vbCrLf
```

```
         KMLText.AddText vbTab & "<description>" & CStr(record.Data("Description")) &
"</description>"  & vbCrLf
         KMLText.AddText vbTab & "<name>" & CStr(record.Data("Name")) & "</name>"  &
vbCrLf
         KMLText.AddText vbTab & "<LookAt>" & vbCrLf
         KMLText.AddText vbtab & vbtab & "<longitude>" & CStr(record.Data("Longitude
(I)")) & "</longitude>"  & vbCrLf
         KMLText.AddText vbtab & vbtab & "<latitude>" & CStr(record.Data("Latitude
(I)")) & "</latitude>"  & vbCrLf
         KMLText.AddText vbtab & "</LookAt>" & vbCrLf
         KMLText.AddText vbtab & "<TimeStamp>" & vbCrLf
         KMLText.AddText vbtab & vbtab & "<when>" & CStr(record.Data("WhenYr")) & "-"
& CStr(record.Data("WhenM")) & "-" & CStr(record.Data("WhenD")) & "</when>" & vbCrLf
         KMLText.AddText vbtab & "</TimeStamp>" & vbCrLf
         KMLText.AddText vbtab & "<Point>" & vbCrLf
         KMLText.AddText vbtab & vbtab & "<coordinates>" & CStr(record.Data("Longitude
(I)")) & "," & CStr(record.Data("Latitude (I)")) & "</coordinates>" & vbCrLf
         KMLText.AddText vbtab & "</Point>" & vbCrLf
         KMLText.AddText "</Placemark>"  & vbCrLf

      next

      KMLText.AddText "</Folder>"  & vbCrLf
      KMLText.AddText "</kml>"

End Sub
```

IMS

IMS Selections from Queries not being Updated/Refreshed

http://69.17.46.171/Site/Thread.aspx?id=17271&ti=632708588838600000

Also see:
http://www.georeference.org/Forums/tabid/71/view/topic/forumid/5/postid/8314/Default.aspx

bnhilton at 12/22/2005 12:10 AM (#17328)

Hi Adam,

Maybe I wasn't clear. Let me explain again.

The query works correctly – user draws an area, of a specified size, on the screen and then clicks a query button to select all the objects contained within that area. This works fine – all the information is returned in the table. As defined by the query:
SELECT * FROM [My Table], [TrackArea Table] WHERE Contains([TrackArea Table].[ID], [My Table].[ID]).

I understand this is not a parameter query; I do need it to be, just want to return all the information regarding the objects contained within the user-defined area.

Yes, if they pan or zoom, the results of the query go away – this is not the problem.

The problem occurs when the user draws a new area on the screen and then clicks the query button to select all the objects contained within that area. The results of the previous query are returned (and yes, I have checked, there are objects within the user-defined area that should be selected).

For some reason the query is not "cleared" after the initial run.

Thanks for your help,

Brian

adamw at 12/22/2005 6:02 AM (#17332)

Got it.

Replace the call to MapServer.Query with:

```
//JScript
Application.Lock;
mapserver.Render();
mapserver.Query("Query Name", "", true, false);
Application.UnLock;

This will select objects under the area, albeit not in the fastest way possible.
```

```
Final code below (LAK)

map.asp


// set render parameters for the TrackArea drawing
var wktArea = Session("wktArea");
if (wktArea == null || wktArea == "")
wktArea = " "; // yields a NULL geom
mapserver.RenderParameters("TrackArea") = wktArea;

//Added this line
//JScript
Application.Lock;
mapserver.Render();
mapserver.Query("Query Name", "", true, false);

Response.BinaryWrite(mapserver.Render());
}
```

Manipulating IMS Layers via Code

http://69.17.46.171/Site/Thread.aspx?id=24275&ti=632883409638900000

exactmike at 7/12/2006 10:07 PM (#24276)

I think I've got a pretty good handle on how the layers code in IMS can be manipulated from reading through the code in default.asp, the GISAdvisor IMS Template, and from several posts on this forum, listed here for convenient access for anyone else reading this post and wanting to learn about layers in IMS: (*I have added the links below – L. Ketch*)

Layer Code (turn on/off) IMS
http://69.17.46.171/Site/Thread.aspx?id=23644&ti=632881130615630000

How to have some layers unchecked when a webpage is loaded?
http://69.17.46.171/Site/Thread.aspx?id=22635&ti=632862969446400000

Set layer to layer off
http://69.17.46.171/Site/Thread.aspx?id=16433&ti=632677302410000000

Published layers
http://69.17.46.171/Site/Thread.aspx?id=10545&ti=632496579000000000

How to disable layers on startup
http://69.17.46.171/Site/Thread.aspx?id=3817&ti=632272818600000000

My question is, what does the following line of code do? I understand that it is setting the value of variable layerDefv, but I'm not familiar with what the "?" and the ":" might be doing. The function parameter also uses this syntax or something similar.

```
var layerDefv = (command == "startup")? "on": "";
```

jkelly at 7/12/2006 10:42 PM (#24278)

It is saying that if command == "startup" then layerDefv = "on", otherwise layerDefn = "". As far as I can see, it is a computationally faster way of writing an if else conditional statement, but I am not really all that sure.

James

Layer Code (turn on/off) IMS

http://69.17.46.171/Site/Thread.aspx?id=23644&ti=632881130615630000

nmack at 6/30/2006 6:59 AM (#23645)

When a particular layer is unchecked and "Apply" is clicked, I want to turn off other layers. So, say if 'highways' are turned off, I want highways labels to also turn off.

To do this, I've used the variable layerChck:

```
if (layerList[layerItem] != "") {
layerChck = parameter("layer" + layerSqno, layerDefv);

if (layerChck == "layer21, off") {
  turnOffAerials = "y";
}
if (layerChck == "layer21, on") {
  turnOffAerials = "n";
}
```

I then pass the "n" or "y" to the html, and on reload use the mapserver.TurnLayer (which works) to turn off the desired layers based on what the turnOffAerials variable says.

What am I doing wrong? From the layerChck declaration, I think I have the set the variable values correctly.
adamw at 7/10/2006 7:24 AM (#24079)
The code should probably be:

```
//JScript
if (layerList[layerItem] != "") {
  layerChck = parameter("layer" + layerSqno, layerDefv);
  if (layerChck != "")
    layerChck = "checked";

  if (layerSqno == 21)
    turnOffAerials = (layerChck != "checked");

  ...
}
```

How to have Some Layers Unchecked when a Webpage is Loaded?

http://69.17.46.171/Site/Thread.aspx?id=22635&ti=632862969446400000

3H3X at 6/15/2006 3:09 AM (#22636)

Hi all,

I know it has something to do with:

```
layerChck = parameter("layer" + layerInd, layerDefv);
if (layerChck != "")
layerChck = "checked";
```

but when the webpage is generated it just checks to see whether the layer is present in the project and not whether the layer is visible/not visible. As a result it always displays all the layers as checked (visible) by default.

Any ideas?

Cheers
T

ECUgeog at 6/15/2006 1:37 PM (#22686)

I fumbled around with the same thing, and finally got it working by hard coding the layer names into the code... like this

```
if (layerList[layerItem] != "") {

if ((command == "startup") && (layerList[layerItem] == "Highway Labels" ||
layerList[layerItem] == "City Labels" )){
layerChck = parameter("layer" + layerSqno, "off");
//}else if (layerList[layerItem] == "Eastern NC Counties" || layerList[layerItem] ==
"Extra"){

}else{

layerChck = parameter("layer" + layerSqno, layerDefv);
if (layerChck != ""){
layerChck = "checked";
}
}
layersC = layersC + "<input id=\"layer" + layerSqno + "\" name=\"layer" + layerSqno +
"\" " + layerChck + " type=\"checkbox\">" + encode(layerList[layerItem]) + "\n";
}
```

3H3X at 6/19/2006 2:15 AM (#22852)

Yep, it is working!

However, i have 30 different map-projects on my IMS which means a lot of hardcoding layers!!
BTW I had to do the same thing for dropdown menus too.

Thanks!

adamw at 6/19/2006 6:55 AM (#22881)
You could store the names of layers which should be shown by default in a separate comments component within each MAP file. This way, you would be able to use the exact same ASP code on each of your web sites.

Set Layer to Layer Off

http://69.17.46.171/Site/Thread.aspx?id=16433&ti=632677302410000000

jluchette at 11/15/2005 9:09 AM (#16434)
how would you set a layers visibility off in an ASP page using javascript?

In had the layer visibilty off in my .map file, but the layers are still on after I reload my map.

jluchette at 11/15/2005 9:56 AM (#16435)
wow, it was easier than i thought. i put this code right before "create results" section, and it works like a charm...

```
if (command != "layers"){
   mapserver.turnlayer(1, false);
   mapserver.turnlayer(2, false);
   }
```

now, the only problem is the check boxes in the layers pane don't reflect my changes.

Any ideas how to do that??

jkelly at 11/15/2005 4:40 PM (#16447)
Joe

You should have some code in there that looks similar to the following. I have added in a few extra lines using the if statement to turn the check boxes off on specific layers.

```
[code jscript]

if (layerList[layerItem] != "") {
   if (region != "" && (command == "startup" || status == "hotstart") &&
(layerList[layerItem] == "Cadastral Parcels" || layerList[layerItem] == "Contours")){
    layerChck = parameter("layer" + layerSqno, "off");
   }else{
    layerChck = parameter("layer" + layerSqno, layerDefv);

    if (layerChck != "")
     layerChck = "checked";
   }
```

```
   layersC = layersC + "<tr><td class=\"panetext\"><input id=\"layer" + layerSqno +
"\" name=\"layer" + layerSqno + "\" " + layerChck + " type=\"checkbox\">" +
encode(layerList[layerItem]) + "</td></tr>\n";
  }

[/code]
```

Cheers

James

jluchette at 11/16/2005 9:30 AM (#16468)

beautious!! thanks.

In case anyone is interested, the code i have working is here is in between the backslash lines, the rest was standard output from the ASP template, i can easily add more layers to the list)

```
[code jscript]
for (layerItem in layerList) {
     ////////////////////////////////////////////////////////////////////////
               if (layerList[layerItem] != "") {
                         if ((command == "startup" ) && (layerList[layerItem] ==
"Cities" || layerList[layerItem] == "Highways")){
                            layerChck = parameter("layer" + layerSqno, "off");
                         }else{
     ////////////////////////////////////////////////////////////////////////
                             layerChck = parameter("layer" + layerSqno, layerDefv);
                                if (layerChck != ""){
                                 layerChck = "checked";
                                 }
                             }
                        layersC = layersC + "<tr><td class=\"panetext\"><input
id=\"layer" + layerSqno + "\" name=\"layer" + layerSqno + "\" " + layerChck + "
type=\"checkbox\">" + encode(layerList[layerItem]) + "</td></tr>\n";
                }
              layerSqno++;
          }
        if (layersC != "") {
                 layersC = "\n<table>\n" + layersC +
                         "<tr><td><input type=\"hidden\" id=\"count\" name=\"count\"
value=\"" + layerSqno + "\"></td></tr>\n" +
                         "<tr><td><input type=\"button\" class=\"panetext\"
value=\"Apply\" onclick=\"invokePane('layers');\"></td></tr>\n" +
                         "</table>\n";
        }
}
[/code]
```

Published Layers

http://69.17.46.171/Site/Thread.aspx?id=10545&ti=632496579000000000

AndreCoutinho at 4/18/2005 9:53 AM (#10546)

Hi there. I have a simple question.

In my published project i have 5 layers. I need to have 3 of them permanently on, while the 2 layers left can be switched on/off by the users.

3 always on, the user can't switch them off. In fact I don't even want them to appear on the layer list

2 that can be switched on/off.

Can anyone help me?

Thanks in advance

AndrT Coutinho

PS: i tried to use <input type=hidden.... but that conceals the layers, and i want to see them but not allow the user to switch them off.

AndreCoutinho at 4/20/2005 10:04 AM (#10618)

I managed a way to do this. I understand that my post wasn't too clear, but i solved the problem.

Still, thanks to all that viewed this post.

AndrT Coutinho

KlausDE at 4/20/2005 11:13 AM (#10621)

Please report. What's the solution? Klaus

AndreCoutinho at 4/21/2005 5:25 AM (#10644)

To ensure that you only see the layers you want in the layers panel (this can be used to stop the user from selecting/unselecting certain layers), you need to check if the layerItem (the number of the layer) is the one you want to show on the panel. This way you keep the layers that shouldn't be selected/unselected, out of the layers panel.

```
{code jscript - temp tag to be replaced}

// create layers pane if necessary

for (layerItem in layerList) {
if (layerList[layerItem] != "") {
layerChck = parameter("layer" + layerSqno, layerDefv);
if (layerChck != "")
```

```
layerChck = "checked";

//this ensures we only see in the layers panel, the layers we want

if(layerItem == 1)

//-------------------------------------------------------

layersC = layersC + "<input id=\"layer" + layerSqno + "\" name=\"layer" + layerSqno +
"\" " + layerChck + " type=\"checkbox\">" + encode(layerList[layerItem]) + "\n";

}

layerSqno++;

}
{/code - temp tag to be replaced}
```

The problem I was having, is that when I do this, the layers that don't appear in the panel, also disappear from the map...

The solution for that is to make those layers permanently on, although they don't show on the panel (that was my goal).

So I did this...

```
{code jscript - temp tag to be replaced}

// handle mapserver commands that do not change state string

. . .
else if (command == "layers") {

for (index = 0; index < parseInt(count); index++){

mapserver.TurnLayer(index, parameter("layer" + index, "") == "on");

//keeps the layer 0 on

if(mapserver.TurnLayer(0,true))

mapserver.TurnLayer(0, parameter("layer" + 0, "")== "on");

}

//------------------------------

. . .
{/code - temp tag to be replaced}
```

This is just a small example, I was working with 2 layers only, but it works. I managed to hide layer 0 from the panel, but keeping it always on, allowing the user to only select/unselect the layer 1.

Cheers from Portugal,

AndrT Coutinho

How to Disable Layers on Startup

http://69.17.46.171/Site/Thread.aspx?id=3817&ti=632272818600000000

cas at 7/20/2004 8:37 AM (#3818)

Hi,

How do i disable the layers on startup?

The layers apears enabled when i enter in the web site, but i want that it be disabled and only when the user clicks in the checkbox of a layer, it apears.

Any help on this?

P.S.:I'm working with manifold v. 5.50.

adamw at 7/26/2004 9:38 AM (#3953)

cas - 2004-07-20 5:37 AM - How do i disable the layers on startup? The layers apears enabled when i enter in the web site, but i want that it be disabled and only when the user clicks in the checkbox of a layer, it apears.

Use something like this in DEFAULT.ASP (untested code):

```
{code jscript - temp tag to be replaced}

var state = parameter("state", "");

// create mapserver object

var mapserver;

mapserver = Server.CreateObject("Manifold.MapServer");

mapserver.Create(Server.MapPath("config.txt"), state, Server);

// turn off all layers but the first if just entered the site

if (state == "") {

var layerList = mapserver.Layers.split("\n");

var index = 0;

for (index = 1; index < layerList.length; index++)

mapserver.TurnLayer(index, false);
```

```
}

{/code - temp tag to be replaced}
```

cas at 7/27/2004 6:41 AM (#3984)

Thank you Adam it works fine.

But there is only one more small problem that is:

The layers are disabled on startup and the checkboxes have to apear disabled too.

I solved the problem by changing the variable layerChck = "checked" to layerChck = "unchecked" and the checkboxes apear disabled too.

But what happens when i click in one checkbox, the check apears and turn on the layer but disapears again, i mean the layer is turned on but the checkbox is unchecked.

Can you help me?

Thanks again. :-)

adamw at 7/27/2004 11:08 AM (#3991)

cas - 2004-07-27 3:41 AM - But what happens when i click in one checkbox, the check appears and turn on the layer but disapears again, i mean the layer is turned on but the checkbox is unchecked. Can you help me?

Could you post your code?

cas at 7/30/2004 10:44 AM (#4080)

The code is here:

```
{code - temp tag to be replaced}

// create layers pane if necessary

var layersC = "";

var layerChck = "";

var layerDefv = (command == "startup")? "on": "";

//var layerList = layers.split("\n");

layerList = new Array(9);

layerList[0]=""; //Layer Coordenadas

layerList[1]="Monumentos"; //Layer Monumentos

layerList[2]=""; //Layer Nomes dos Monumentos
```

```
layerList[3]="Hoteis"; //Layer Hoteis

layerList[4]=""; //Layer Nomes dos Hoteis

layerList[5]="Estabelecimentos"; //Layer Estabelecimentos

layerList[6]=""; //Layer Nomes dos Estabelecimentos

layerList[7]=""; //Layer Nomes das Ruas

layerList[8]="Ruas"; //Layer Ruas


var layerSqno = 0;

var layerItem;

for (layerItem in layerList) {

if (layerList[layerItem] != "") {

layerChck = parameter("layer" + layerSqno, layerDefv);

if (layerChck != "")

layerChck = "checked";

if (layerList[layerItem] == "Monumentos")

layersC = layersC + "<input id=\"layer" + layerSqno + "\" name=\"layer" + layerSqno +
"\" " + layerChck + " type=\"checkbox\" onclick=\"invokePane('layers');\">" + "\n" +
"" + "\n" + encode(layerList[layerItem]) + "\n";

if (layerList[layerItem] == "Estabelecimentos")

layersC = layersC + "<input id=\"layer" + layerSqno + "\" name=\"layer" + layerSqno +
"\" " + layerChck + " type=\"checkbox\" onclick=\"invokePane('layers');\">" + "\n" +
"" + "\n" + encode(layerList[layerItem]) + "\n";

if (layerList[layerItem] == "Hoteis")

layersC = layersC + "<input id=\"layer" + layerSqno + "\" name=\"layer" + layerSqno +
"\" " + layerChck + " type=\"checkbox\" onclick=\"invokePane('layers');\">" + "\n" +
"" + "\n" + encode(layerList[layerItem]) + "\n";

if (layerList[layerItem] == "Ruas")

layersC = layersC + "<input id=\"layer" + layerSqno + "\" name=\"layer" + layerSqno +
"\" " + layerChck + " type=\"checkbox\" onclick=\"invokePane('layers');\">" + "\n" +
"" + "\n" + encode(layerList[layerItem]) + "\n";

}

layerSqno++;
```

```
}

if (layersC != "") {

layersC = "\n\n" + layersC +

"<input type=\"hidden\" id=\"count\" name=\"count\" value=\"" + layerSqno + "\">\n" +

"\n";

}

{code - temp tag to be replaced}
```

WillH at 7/30/2004 2:14 PM (#4086)

Instead of the default

```
{code - temp tag to be replaced}

if (layerChck != "")

layerChck = "checked";{/code - temp tag to be replaced}

You could check to see if the IMS is actually displaying the layer and apply the check
to the box appropriately

{code - temp tag to be replaced}

if (mapserver.LayerShown(layerItem) == true)

layerChck = "checked";{/code - temp tag to be replaced}
```

This may require you to also modify the code where you have released the mapserver object.

cas at 8/5/2004 5:51 AM (#4189)

Hi, Will.

It's working fine.

With the code that Adam posted and your code, i solved, not only this problem, but also another problem that i had.

Thank you both.

Labels

Adding Labels via Code

http://69.17.46.171/Site/Thread.aspx?id=18202&ti=632737768954800000

njengler at 1/24/2006 11:50 AM (#18203)

I am trying to add labels programmatically (VB.NET, XP Pro Sp2, Manifold 6.5) to a newly created drawing. This is my code:

```
Dim mLabel As Manifold.Labels
mLabel = mDrawing.Document.NewLabels("Road Labels", mDrawing)
mLabel.Text = "[" + theRoadField + "]"

For i = 0 To mDrawing.ObjectSet.Count - 1
   mLabel.LabelSet.Add(mDrawing.ObjectSet(i), _
      mDrawing.ObjectSet(i).Geom)
Next
```

I get the following error: "Cannot add label"

The label object is created, linked to the drawing. When I open the label window i see a point but no text. If I 'Unlink' the label from the drawing and then open the label window, the text appears.

Does anyone know what I am doing wrong here? I have also tried omitting the second argument in the LabelSet.Add() line without any luck.

adamw at 1/24/2006 10:11 PM (#18231)

I have been able to reproduce this. This is a bug, which will be fixed.

Creating Bound Labels

http://69.17.46.171/Site/Thread.aspx?id=18608&ti=632762469194000000

KlausDE at 2/17/2006 3:05 AM (#19019)

For bound Labels use LSet.Add Object, Geom

```
     ' bind the labels to a table column
Labels.Text = "[LabelColumn]"
For each object In theDrawing.ObjectSet
  Labels.LabelSet.Add object, object.Geom
Next
```

NOTE: SEE ADAM'S LAST COMMENT IN THE NEXT TOPIC:
"Bound Labels Created by Script do Not Display"

You can bypass the manual adding of each layer by setting the Synchronized property to False and then back to True: In the code above, forget the For loop and just use two lines:

```
Labels.Synchronized = False
Labels.Synchronized = True
```

AND,

V7 supports a new extentions to the Document Object
NewLabels(String name, Drawing drawing, Boolean exactName, **Boolean populate**)

When the populate parameter is set to True, the system automatically populates the labels component with a label for each drawing object.

Bound Labels Created by Script do Not Display

http://69.17.46.171/Site/Thread.aspx?id=22339&ti=632854363953230000

Lorne at 6/8/2006 4:36 AM (#22340)

The VBscript below assigns the column name [Labels_E] to the Text property of the Labels_E object. The Label component is created fine and the correct field is shown in the Labels|Text menu dialog. However, the labels do not appear when the component is opened. I can change the column using the Labels|Text menu but the labels still do not appear. The labels immediately display if I save the project. I don't see a Labels Refresh method or anything similar. What am I missing?

Thanks
Lorne

```
Document.NewLabels Temp_Str & "_Labels_E", Points, TRUE
Set Labels_E = Comps.Item(Comps.Count - 1)
Labels_E.Text = "[Labels_E]"
Labels_E.Synchronized = TRUE
Labels_E.PerLabelFormat = TRUE
.... etc.
```

Lorne at 6/8/2006 10:24 AM (#22384)

Sorry for the post. I have to add each label object to Labels_E.LabelSet.

```
For Counter = 0 To Points.ObjectSet.Count - 1
Set Label_Point = Points.ObjectSet(Counter)
Labels_E.LabelSet.Add Label_Point, Label_Point.Geom
Next
```

Just for interest, why does Manifold automatically add the labels for you when you Save the project.

Lorne

adamw at 6/9/2006 5:59 AM (#22446)

It does this because you have set the labels to be synchronized with the drawing.

You could avoid adding labels by hand by setting the Synchronized property to False and then back to True:

```
'VBScript
Sub Main
  Set Points = Document.ComponentSet("Drawing")
  Set Labels_E = Document.NewLabels("Labels_E", Points)
  Labels_E.Text = "[ID]"
  Labels_E.Synchronized = False
  Labels_E.Synchronized = True
  Labels_E.PerLabelFormat = True
End Sub
```

I have added a request to populate the labels component the moment you set the Synchronized property to True for the first time so that there is no need to set it to False first.

adamw at 6/9/2006 5:59 AM (#22447)

A small note:

You cannot rely on the added labels component to be the last one in the component set. Use the value returned by Document.NewLabels:

```
'VBScript
Set Labels_E = Document.NewLabels(Temp_Str & "_Labels_E", Points, TRUE)
```

Lorne at 6/9/2006 7:22 AM (#22460)

Sorry Adam, one additional question.

There are many examples on GeoReference and the archive site of using my previous method to assign a new component Pasted from the clipboard to an object variable.

```
Set Comps = Document.ComponentSet
DrawingObj.Copy TRUE
Document.PasteAs ComponentDrawing
Set NewDrawing = Comps.Item(Comps.Count - 2)
```

Is a newly Pasted component guaranteed to be the last in the component set? This would be rather important when attempting to duplicate a component or a selection from a component via script since there does not appear to be a DUPLICATE method for any of the component types.

Thanks again
Lorne

adamw at 6/9/2006 7:53 AM (#22465)

There is never such a guarantee.

One of the possible ways to locate the components created by the paste operation (which can create several components!) is to save the IDs of all existing components prior to doing the paste and locate the new components after the paste. Of course, this is too much code for something that has to be done so frequently, so things will likely improve in the future.

Script to Place Labels on Point Objects

http://69.17.46.171/Site/Thread.aspx?id=18608&ti=632762469194000000

nujseyer at 2/22/2006 12:09 AM (#19103)

Here is my complete script for labeling... Somehow, somewhere I still can't add the label.

```
Sub Main
Set document = Application.ActiveDocument
Set components = document.ComponentSet
Set theDrawing = components(components.ItemByName("PlotXY"))

' -- obtain source data readers
Set data = components(components.ItemByName("MyPosition"))
Set dataColumns = data.ColumnSet
Set dataRecords = data.RecordSet

'Get value of first record
Set dataRecord = dataRecords(0)
Set dataColumn = dataColumns(0)

'Remove CP Labels for testing
'------------------
i = 0
Do
    if cstr(components(i).name) = "CP Labels" Then
    components.Remove(i)
Else
    i = i + 1
    End If
Loop Until i = components.Count
'------------------

' -- create the labels component
Set Labels = Application.ActiveDocument.NewLabels("CP Labels", theDrawing, true)

' bind the labels to a table column
' Labels.Text = dataRecord.Data("Long")   ' Long is a column in MyPosition Table

' This is done incorrectly above… see note from Klaus below
Labels.Text= dataRecord.Data("[Long]")   ' Long is a column in MyPosition Table


For each object In theDrawing.ObjectSet
'/-------------------------------------------------------
    Labels.LabelSet.Add object, object.Geom
'/-------------------------------------------------------
```

```
Next
Labels.Open()

End Sub
```

KlausDE at 2/22/2006 12:24 AM (#19104)

Put a string into Labels.Text exactly like you would do manually in the Label-Edit Dialog. Manifold evaluates this string for you and performs all the record fetching, type conversions and string concatening:

```
Labels.Text= "[Long]"
```

Adding Labels at x/y Points to an Unbound Label Component

http://69.17.46.171/Site/Thread.aspx?id=20662&ti=632816745731870000

mikem at 4/25/2006 9:13 AM (#20663)

Specifically, I want to read a text string, with associated x and y co-ordinate fields, from a table and then place the text, at the appropriate xy location, in the unbound Label component.

Here's a piece of my code - a mess, I know, and I've forgotten how to bracket it.

```
Set labelGeom = Application.NewGeom()
labelGeom.Branchset = objectPoint
set bset = labelGeom.Branchset
Set labSet = labelcomponent.Labelset
labSet.Add("225", bset)
```

KlausDE at 4/26/2006 12:02 AM (#20677)

Two or three problems here:

1. Don't use the name of a Constant 'objectPoint' for a variable.
2. Omit the brackets surrounding the parameters of the LabelSet.Add function. (To be honest - again and again I find this by trial and error running into the Syntax error.)
3. You MUST assign a GeomType to NewGeom() but you do not need this detouring here.

So the code with a dummy point simply reads

```
--VBScript
Set app = Application
Set comps = app.ActiveDocument.ComponentSet
Set labelcomponent = comps("Labels Comp")
Set thePoint = app.NewPoint(10,10)

Set labSet = labelcomponent.LabelSet
labSet.add "255", thePoint
```

Set Label Rotation Based upon a Column Value

http://69.17.46.171/Site/Thread.aspx?id=5682&ti=632346591000000000

WillH on 10/19/2004 11:34 AM (#5690)

….. Here is a small script that will set the rotation to the associated value in a column (in other words, first set the formatting to unique values and choose the proper column, close the dialog then run the script.

```
Option Explicit

Sub Main
    ' Modified for V7 (LAK)
  Dim Comps, Drwg, fVS, fValObj

  Set Comps = Document.ComponentSet
  Set Drwg = Comps("Drawing")
  Set fVS = Drwg.PointRotation.Values

  For Each fValObj in fVS
    fValObj.Formatting = fValObj.Value
  Next

End Sub
```

Layouts

http://forum.manifold.net/Site/Thread.aspx?id=32835

Applying a Layout-Template to Different Layouts

pcardoso on 12/29/2006 10:29 AM (#32836)

I don"t know how to solve this:
I have a number of layouts to do, based in a common template, created from a model created using a map (MAP1). I stored the template as xml.

In order to reproduce exactly the template into another map (MAP2), with a different set of drawings, I tried both: select from Menu Edit/Template/Apply... and choose the template from MAP1 and Edit/Template/Apply file... loading the xml file.

By doing this to MAP2 the reference changes to MAP1. Well, this is not interesting.

Is this supposed to work like this? Is there any way to keep everything from a template but DO NOT change the MAP reference?

mdsumner on 1/1/2007 3:57 AM (#32863)

There's no way to provide a generic reference to a component - it needs an explicit named component in the template text.

Just change the text in the XML to the map name you want and it will work - this would be pretty easy to semi-automate if it matters that much. I can't see an easy way that you could reference a component without the explicit name - it generates too many possibilities. How to choose which map?

Automate Layout Creation

http://69.17.46.171/Site/Thread.aspx?id=19428&ti=632773629605470000

eandelin at 3/6/2006 8:23 PM (#19429)

I have a project that will involve extracting and adding multiple layers, such as roads, schools, commercial businesses, and labels to +/- 900 sq mi area. The final maps will then be cut according to a pre-determined grid into individual tiles and exported for printing. If I create the map with all the labelling and layers, then place a grid on it, is there a way to automate the layout creation as opposed to doing a clip with intersect on each individual tile?

seatrails at 3/7/2006 7:32 PM (#19472)

We create a lot of irregular grids for our maps that need to be reconciled to layouts and exported as tiffs or pdfs. I include a script below that hacks together some of the scripts. It iterates through the Tiles Drawing and generates a layout entry for each object. I'll leave it to you to figure out

the appropriate scale and size. The tiles drawing can be created on a regular grid simply by using the show gird-create tiles function.

Tiles is a drawing with the desired grid (one tile = one area object). If you have a regular grid you can use the create tiles grid function to generate tile objects with which you can derive the box corner min/max values.

Make sure the Tiles drawing and the map component have the same projection and both local values are equal to 1.0. This assumes meters. Make sure your default printer preferences are set to your desired output page size. I also include an export routine you can hack to desired effect.

```
' ************************************************
Sub Main

   SET ManDoc = Application.activedocument
   SET ManComps = ManDoc.ComponentSet
   Set TilesObjs = ManComps("Tiles").ObjectSet

   in_borderMargin = 10

   in_scale = 30000
   meters_inch = 1609.27
   meters_pica = meters_inch / 72 / 63360 * in_scale

   TileIndex = 1

   For Each TilesObj in TilesObjs
      SET ManLayout = ManDoc.NewLayout("Tile_"+Cstr(TileIndex))

      ManLayout.EntrySet.AddComponent(ManComps("Map"))
      SET ManLE = ManLayout.EntrySet.Item(0)

      BoxWidth = in_borderMargin * 2 + TilesObj.Geom.Box.Width / meters_pica
      BoxHeight = in_borderMargin * 2 + TilesObj.Geom.Box.Height / meters_pica

      With ManLE
         .Border = LayoutBorderCoordinatesGraticule
         .BorderDegMinSec = True
         .BorderMargin = 10
         .BorderRounding = -3
         .Scope = LayoutScopeBox
         .ScopeArea = TilesObj.Geom.Box
         .ControlPoints = LayoutStateOff
         .Grid = LayoutStateOff
         .Legend = LayoutStateOff
         .NorthArrow = LayoutStateOff
         .ScaleBar = LayoutStateOff
         .Background = LayoutStateOff
         .Graticule = LayoutStateOn
         .MoveTo 0,0, BoxWidth, BoxHeight
      End With

   TileIndex = TileIndex + 1
   Next
```

```
   application.messagebox "Layouts complete"
End Sub
' ************************************************

************Export Layouts code
Sub Main

turnOffGratBorder = false
set apptrans = Application.NewColor("Trannie", 0, 0, 1)
set appblack = Application.NewColor("Whitie", 0, 0, 0)
SET ManDoc = Application.activedocument
set oExportPDF = application.NewExport("PDF")
oExportPDF.Compression = False
oExportPDF.Resolution = 400
oExportPDF.VectorResolution = 400
oExportPDF.Transparency = True

SET ManComps = ManDoc.ComponentSet

for each ManComp in ManComps
   'if ManComp.Type = ComponentLayout then
   if ManComp.Type = ComponentLayout and ManComp.Name = "CA005" then

      if turnOffGratBorder then
        For each vEntry in ManComp.EntrySet
          With vEntry
            .Graticule = LayoutStateOff
            .BorderColor = apptrans
          End With
        Next
      end if

      with oExportPDF
      .PageFilter = "1"
      .Export ManComp, "C:\STWork\BayVectors\" + ManComp.Name +"A.pdf", PromptNone
      .PageFilter = "2"
      .Export ManComp, "C:\STWork\BayVectors\" + ManComp.Name +"B.pdf", PromptNone
      end with

      if turnOffGratBorder then
         For each vEntry in ManComp.EntrySet
            With vEntry
              .Graticule = LayoutStateOn
              .BorderColor = appblack
            End With
         Next
      end if

   end if

   Next

   application.messagebox "Processing Complete"

end sub
```

Custom Layout Templates

http://69.17.46.171/Site/Thread.aspx?id=26722&ti=632928851816070000

kgf at 8/30/2006 1:28 PM (#26723)

Hi all. Can anyone help with custom templates?. I have layout templates working fine either from Apply File or from adding to the Config folder. However the problem I have is that they seem to be bound to the name of the Map they were created from. So if I create a template using Map1 and then apply a template to a layout attached to Map2 the template uses Map1. The workaround is to create a layout using default name Map each time, but is there a way to edit the xml file to make the layout refer to the active Map rather than the name of the Map the template was created with?.

Thanks

adamw at 9/2/2006 6:20 AM (#26879)

Use layouts bound to components and use < body ... > instead of < component name=... >.

kgf on 9/3/2006 12:59 PM (#26957)

Thanks very much, I changed

```
<component background="auto" border="thin" ....
```

to

```
<body background="auto" border="thin" .....
```

and all seems to work now.

Lines

Line Intersects Line at Coordinates?

http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/2436/Default.aspx#2440
http://forum.manifold.net/Site/Thread.aspx?id=2824&ti=632218042200000000

(Also see the heavily modified version that follows (Find Line Interest Coordinates – L. Ketch)

mdsumner on 6/2/2004 8:17 PM (#2829)

```
Sub Main
   Set doc = Application.ActiveDocument
   Set comps = doc.ComponentSet
   Set comp = comps("Drawing")

   ' get the objects in the drawing - assuming all are lines here
   Set objects = comp.ObjectSet

   ' get the intesection points
   Set pts = objects.GeomSet.IntersectionPoints()

   ''''' is precision an issue?:
   ''''' eps = 0.1
   ''''' 'Set pts = objects.GeomSet.IntersectionPoints(eps)

     ' report

   Application.Messagebox  pts.count

   Set aPt = pts.Item(0).Center

   Application.Messagebox "X: " & aPt.X & vbcrlf & "Y: " & aPt.Y
End Sub
```

Find Line Intersect Coordinates

L. Ketch – Highly modified version of Mike Sumner's code obtained from:
http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/2436/Default.aspx#2440

' thinpig 06/02/2004 5:31 PM Alert
' Which object or method can I call, in order to get the X,Y coordinate
' of two lines which are intersected..
' It can be easily done by using Transform tool bar, but how can I apply it by script??

' mdsumner 06/02/2004 8:17 PM Alert
' This script assumes that your "Drawing" has some lines in it,
' some of which intersect:
' It first reports the number of intersections (the "count" property
' of the GeomSet returned by "IntersectionPoints"), and then the

```
' coordinates of the first point in that set.
'
'Sub Main
'  Set doc = Application.ActiveDocument
'  Set comps = doc.ComponentSet
'  Set comp = comps("Drawing")
'
'    ' get the Objs in the drawing - assuming all are lines here
'  Set Objects = comp.ObjectSet
'
'    ' get the intesection points
'  Set pts = Objects.GeomSet.IntersectionPoints()
'
'     """ is precision an issue?:
'     """ eps = 0.1
'     """ 'Set pts = Objs.GeomSet.IntersectionPoints(eps)
'
'    ' report
'  Application.Messagebox pts.count
'  Set aPt = pts.Item(0).Center
'  Application.Messagebox "X: " & aPt.X & vbcrlf & "Y: " & aPt.Y
'End Sub

' ********************************************************************

Option Explicit

'  Author: Lorne Ketch
'  NOTE: Highly modified version of Mike Sumner's code obtained from:
'
http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/2436/Default.
aspx#2440

' ********************************************************************

Sub Main

  Dim Comp, Comps
  Dim Pts, Apt, Objs, Counter
  Dim Report
  Dim ReportWin, Comment_Intersections
  Dim Comment
  Dim NumberOfLines, NumberOfPoints
  Dim PointDrawing, PointObjects

  Set Comps = Document.ComponentSet
```

```
Set Comp = Comps("Drawing")

  ' get the Objects in the drawing - assuming all are lines here
Set Objs = Comp.ObjectSet

If Objs.Count = 0 Then
   Application.Messagebox "No Objects in " & Comp.Name
   Exit Sub
End If

NumberOfLines = 0
For Counter = 0 to  Objs.Count-1
  If Objs.Item(Counter).Type = ObjectLine Then
      NumberOfLines = NumberOfLines + 1
  End If
Next

If NumberOfLines = 0 Then
   Application.Messagebox "No Line Objects in " & Comp.Name
   Exit Sub
End If

  ' Get the intesection points.
  ' NOTE: If precision is an issue, pass the precision
  '       specification to the IntersectionPoints method.
  '       Example:
  '       eps = 0.1
  '       Set pts = Objs.GeomSet.IntersectionPoints(eps)

Set Pts = Objs.GeomSet.IntersectionPoints()

   ' There is really no error handling methodolgy in VBscript.
   ' If no Line Intersections are found in "Drawing" then
   ' the Pts.Count below will generate an error.  The only
   ' way to check this is to set "On Error Resume Next",
   ' check the Point count, then check for an error.
   ' On Error Goto 0 turns off error checking.
On Error Resume Next
NumberOfPoints = Pts.Count
If Err > 0 Then
   Application.Messagebox "There are no Line Intersections in " & Comp.Name
   Exit sub
End If

If  Component_Exists("Point Coordinates") Then
  Set Comment_Intersections = Comps(Comps.ItemByName("Point Coordinates"))
```

```
  Else
     ' Create a Comment
    Set Comment_Intersections = Document.NewComments ("Point Coordinates", TRUE)
  End If

  If  Component_Exists("IntersectionPoints Drawing") Then
     ClearDrawing "IntersectionPoints Drawing"
     Set PointDrawing = Comps(Comps.ItemByName("IntersectionPoints Drawing"))
  Else
     Set PointDrawing = Document.NewDrawing ("IntersectionPoints Drawing",
Comp.CoordinateSystem , TRUE)
  End If

  Set PointObjects = PointDrawing.ObjectSet

  Comment_Intersections.Clear

  For Counter = 1 to NumberOfPoints
    Set aPt = pts.Item(Counter-1).Center
    Comment_Intersections.AddText "Point " & Counter & "   X: " & FormatNumber(aPt.X, 15, -
1) & "  Y: " &  FormatNumber(aPt.Y, 15, -1) & vbcrlf

      ' Add a point at the intersection.
    PointObjects.Add(Application.NewGeom(GeomPoint, Application.NewPoint(aPt.X, aPt.Y)))

  Next

  If Comment_Intersections.Text = "" Then
     Application.messagebox "No Intersecting Lines."
  Else
     Comment_Intersections.Open()
  End If

End Sub

' ****************************************************************

Private Function Component_Exists(ByVal ComponentName)
   Dim Component_Index
   Dim Components

   Set Components =Application.ActiveDocument.ComponentSet

   Component_Index = Components.ItemByName(ComponentName)
   If Component_Index < 0 Then
     Component_Exists = False
```

```
  Else
    Component_Exists = True
  End If

End Function

' ********************************************************************

Sub ClearDrawing (byRef ComponentName)
  Dim Doc, Comp, Comps
  Dim Objs, Counter

   ' This is a modified version of Mike Sumner's code obtained from:
   '
http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/2436/Default.
aspx#2440

  Set Doc = Application.ActiveDocument
  Set Comps = Doc.ComponentSet
  Set Comp  = Comps(ComponentName)
  Set Objs  = Comp.ObjectSet

  For Counter = Objs.Count-1 to 0 Step -1
    Objs.Remove(Counter)
  Next
End Sub

' ********************************************************************
```

Plotting Points Equidistant along a Line - Script

Approach 1

Bob Heiztman
01/19/2004 3:43 PM
http://www.georeference.org/Forums/tabid/71/forumid/7/tpage/1/view/topic/postid/936/Default.aspx#936

Here is a script I developed as an exercise in plotting points along a line. It will develop "milestones" along a line. See script for details on user settings.

```
' Manifold 5.5 SP2
' Author: Bob Heiztman Jan 2004
'
' Script will add points along single SELECTED line in the active window
' Active window should contain a projected drawing
' User should edit script to provide the distance between mileposts
```

```
Option Explicit

' ******************************************************


Sub Main
   Dim drawingIn, drawingOut, objects, object
   Dim mpLength, legLen, branch, i, s, pt1, pt2
   Dim lenTo, angleTo, distance, postPt, fromPt, bal

      ' user settings
   mpLength = CDbl(1609) ' distance between mile posts in meters

      ' get active drawing window
   Dim doc, windows, window, comps

   Set doc = Application.ActiveDocument
   Set comps = doc.ComponentSet

   If True Then ' True = use active window
      Set windows = Application.WindowSet

      If Windows.Count = 0 Then
         Application.Messagebox "No active component"
         Exit Sub
      End If ' -- fail If there are no opened windows

      Set window = Windows.ActiveWindow
      Set drawingIn = Window.Component

      If drawingIn.Type = ComponentMap Then ' drill down to active layer
         Set drawingIn = Window.ActiveComponent
      End If

      If Not (drawingIn.TypeName = "Drawing" or drawingIn.TypeName="Drawing") Then
         Application.Messagebox "Invalid active component"
         Exit Sub
      End If

   Else ' False - hardcoded name for testing

      Set drawingIn = comps("A")

   End If

   s = InputBox("Drawing to add mileposts to (must exist):",,drawingIn.Name &
"Mileposts")
   Set drawingOut = Application.ActiveDocument.ComponentSet(s)

      ' get selected objects
   Set objects = drawingIn.selection

   If objects.count = 0 Then
      Set objects = drawingIn.ObjectSet
```

```
   End If

   If objects.Count <> 1 Then
      Application.Messagebox "This script can only operate on one line at a time."
      Exit Sub
   End If

   Set Object = Objects(0)
   If Object.TypeName <> "Line" Then
      Application.Messagebox "This script can only operate on one line at a time."
      Exit Sub
   End If

   Set pt2 = object.Geom.BranchSet(0).PointSet(0)
   drawingOut.ObjectSet.Add(Application.NewGeom(GeomPoint, pt2)) ' start post

   legLen = CDbl(0) ' distance remaining to next milepost

   For Each Branch In Object.Geom.BranchSet
      For I = 1 to Branch.PointSet.Count - 1

         Set pt1 = pt2
         Set pt2 = Branch.PointSet(i)
         lenTo = pt1.DistanceTo(pt2)

         If legLen + lenTo < mpLength Then
            legLen = legLen + lenTo
         Else ' next milepost in this segment
            angleTo = CalcAngleRadians(pt1, pt2)
            Set postPt = pt1 ' prime
            bal = lenTo

            Do While bal + legLen > mpLength
               distance = mpLength - legLen
               legLen = 0
               Set fromPt = postPt
               Set postPt = Application.NewPoint
               postPt.X = fromPt.X + distance*Sin(angleTo)
               postPt.Y = fromPt.Y + distance*Cos(angleTo)
               drawingOut.ObjectSet.Add(Application.NewGeom(GeomPoint, postPt))
               bal = bal - distance
            Loop

            legLen = bal - legLen ' record remaining distance

         End If

      Next ' i
   Next

   Application.Messagebox "done!"

End Sub

' *******************************************************

Function CalcAngleRadians (ByRef FromPoint, ByRef ToPoint)
```

```
   Dim Rads, Degs

   If FromPoint.Y = ToPoint.Y And FromPoint.X < ToPoint.X Then
      Degs = 90
   ElseIf FromPoint.Y = ToPoint.Y And FromPoint.X > ToPoint.X Then
      Degs = -90
   ElseIf FromPoint.Y = ToPoint.Y Then
      Degs = 0 ' points are coincident
   Else
      Rads = Atn((ToPoint.X - FromPoint.X) / (ToPoint.Y - FromPoint.Y))
      Degs = Rads * 180 / 3.1415926536 ' convert to degrees

      If FromPoint.Y > ToPoint.Y Then
         Degs = Degs - 180 ' expand to four quarters
      End If

      Do While Degs < -180
         Degs = Degs + 360 ' protect from underflow
      Loop

      Do While Degs > 180
         Degs = Degs - 360 ' protect from overflow
      Loop
   End If

   CalcAngleRadians = Degs * 0.01745329252 ' return radians

End Function

' ********************************************************
```

Approach 2

willh at 4/29/2006 6:00 PM (#20736)
http://69.17.46.171/Site/Thread.aspx?id=20702&ti=632819309105000000

It is amazing how often this comes up. Here is a version I wrote ages ago when I was using the 6.50 beta. It should probably be refactored and maybe even turned into an addin given its popularity. Eh, maybe when I find some time...

```
Sub Main
   Set app = Application
   Set doc = app.ActiveDocument
   Set comps = doc.ComponentSet

   drawingName = app.InputBox("Enter drawing name","","Drawing")
   dist = app.InputBox("Enter distance spacing (native units)","","100")

   Set linesQuery = doc.NewQuery("Temp1", false)
   Set bLengthQuery = doc.NewQuery("Temp2", false)
   Set pointQuery = doc.NewQuery("Temp3", false)

   linesQuery.Text = _
     "SELECT [ID], [Branches (I)] " & VbCrLf & _
```

```
    "   FROM [" & drawingName & "] " & VbCrLf & _
    "   WHERE [Type (I)] IN (2);"


  bLengthQuery.Text = _
    "Parameters [lID] int, [bID] int; " & VbCrLf & _
    "SELECT LENGTH(BRANCH([lID],[bID])) " & VbCrLf & _
    "   FROM [" & drawingName & "] " & VbCrLf & _
    "   WHERE [ID] = lID;"


  pointQuery.Text = _
    "Parameters [lID] int, [bID] int, [dist] int;
    "INSERT INTO [" & drawingName & "] ([geom (i)]) " & VbCrLf & _
    "   (SELECT LINEPOINT(BRANCH([lID],[bID]),[dist]) " & VbCrLf & _
    "    FROM [" & drawingName & "] " & VbCrLf & _
    "    WHERE [ID] = lID);"

  For each rec in linesQuery.Table.RecordSet
     For x = 0 to rec.DataText("Branches (I)") - 1
        bLengthQuery.ParameterSet.Item(0).Value = rec.DataText("ID")
        bLengthQuery.ParameterSet.Item(1).Value = x
        bLength = bLengthQuery.Table.RecordSet.Item(0).Data("Column")

        For j = 0 to Int(bLength/dist) - 1
           pointQuery.ParameterSet.Item(0).Value = rec.DataText("ID")
           pointQuery.ParameterSet.Item(1).Value = x
           pointQuery.ParameterSet.Item(2).Value = dist * (j + 1)
           pointQuery.Run()
        Next
     Next
  Next

  comps.Remove(linesQuery)
  comps.Remove(bLengthQuery)
  comps.Remove(pointQuery)
End Sub
```

Find Coordinates of Inflection Points along a Line

http://www.georeference.org/Forums/tabid/71/forumid/2/tpage/1/view/topic/postid/2074/Default.aspx#2075

mdSumner

04/21/2004 5:24 AM

Shows how to interrogate a line point-set and return values.

```
Sub Main
Set doc = Application.ActiveDocument
Set comps = doc.ComponentSet
Set comp = comps("Drawing")
Set objSet = comp.ObjectSet

Set obj = objSet(0)
```

```
If obj.Type = ObjectLine Then
   Set branch = obj.Geom.BranchSet.Item(0)
   Set pts = branch.PointSet
   Application.Messagebox "This many points in point set: " & pts.Count
   Application.Messagebox "Coordinates of first point:" & vbcrlf & "X: " & _
        CStr(pts(0).X) & vbcrlf & "Y: " & CStr(pts(0).Y)
End If
End Sub
```

Creating Track with different point format for start and end (interesting code for making lines, etc.)

Rusty

03/15/2005 7:52 PM

http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/8000/Default.aspx#8000

```
' ************************************************************

Sub Main

   dim strSQL

   '-- Create the SQL statement for the chosen shark
   strSQL = "Select [SGT-Date], [SGT-Latitude]*-1 AS Latitude, [SGT-Longitude], [SGT-
ID] "
   strSQL = strSQL & "FROM [T-SightingDetails] "
   strSQL = strSQL & "WHERE Shrk_ID = 505 "
   strSQL = strSQL & "ORDER BY [SGT-Date]"
   '-- application.messagebox strSQL

   '-- Use the SQL statement in creating a query with positions and dates for "Bomber"
   Set components = Application.ActiveDocument.ComponentSet

   SET Query = Application.ActiveDocument.NewQuery("Bomber")
   query.Text = strSQL

   '-- Make a table for the data from the query
   SET Table = query.table

   '-- Call CreateLines to join the lines
   CreateLines Table
   '-- Call TaggingOp to mark the position of tagging
   TaggingOp
   '-- Call LastKnownPos to mark the last known position of the shark
   LastKnownPos

END SUB

' ************************************************************

SUB CreateLines (tbl)
   '-- Create lines from the table of records for the selected shark
```

```
   '--Ensure at least two records
   SET records = tbl.Recordset
   IF records.Count < 2 Then
   '-- remove the temperary query component

Application.ActiveDocument.ComponentSet.Remove(Application.ActiveDocument.ComponentSet
.ItemByID(query.ID))
   Application.Messagebox "There should be at least two selected points"
   END IF

   '-- Set up a coordinate system and create a new drawing
   SET CS = Application.NewCoordinateSystem("Latitude / Longitude")
   SET nDwg = Application.ActiveDocument.NewDrawing("Bomber-PTT: 52526", CS)

   '-- Create new point set object
   SET pointset = Application.NewPointSet

   '-- Scan queried table adding locations to point set
   FOR recordIndex = 0 TO records.Count-1
   SET record = records(recordIndex)
   SET point = Application.NewPoint
   '-- Edit: use the query table's column names
   point.X = CDbl(record.data("SGT-Longitude"))
   point.Y = CDbl(record.data("Latitude"))
   pointSet.Add point
   NEXT

   '-- Create a new geometric entity
   SET geom = Application.NewGeom(GeomLine, PointSet)

   '-- Create new line object
   nDwg.ObjectSet.Add geom


   '-- Remove the temperary query component
   Application.ActiveDocument.componentset.Remove("Bomber")

   SET map = Application.ActiveDocument.ComponentSet("Map")

   '-- assign a custom colour to the line for Bomber
   nDwg.LineBackground.DefaultValue.Formatting=application.NewColor("DarkGreen",
0,125,0)
   nDwg.LineForeground.DefaultValue.Formatting=application.NewColor("DarkGreen",
0,125,0)

   SET nLayer = Application.ActiveDocument.NewLayer(nDwg)
   map.LayerSet.Add nLayer
   map.open

END SUB

' ************************************************************

SUB TaggingOp ()

   '-- get the coordinates of the first point in the line
   SET doc = Application.ActiveDocument
```

```
   SET comps = doc.ComponentSet
   SET comp = comps("Bomber-PTT: 52526")
   SET objSet = comp.ObjectSet

   SET obj = objSet(0)
   IF obj.Type = ObjectLine THEN
   SET branch = obj.Geom.BranchSet.Item(0)
   SET pts = branch.PointSet
   END IF
   '-- Set up a coordinate system and create a new drawing
   SET CS = Application.NewCoordinateSystem("Latitude / Longitude")
   SET nDwg = Application.ActiveDocument.NewDrawing("Start-Bomber", CS)

   SET pointset = Application.NewPointSet
   SET point = Application.NewPoint

   '-- Edit: use pts.Count method to set the x & y coords for first point on line
   point.X = CDbl(pts(0).x)
   point.Y = CDbl(pts(0).y)
   pointset.Add point

   '-- Create a new geometric entity
   SET geom = Application.NewGeom(GeomPoint, PointSet)

   '-- Create new point object
   nDwg.ObjectSet.Add geom
   SET map = Application.ActiveDocument.ComponentSet("Map")
   SET nLayer = Application.ActiveDocument.NewLayer(nDwg)

   map.LayerSet.Add nLayer

   '-- assign a custom colour to the tagging operation for Bomber
   nDwg.PointBackground.DefaultValue.Formatting=application.NewColor("black", 0,0,0)
   nDwg.PointForeground.DefaultValue.Formatting=application.NewColor("black", 0,0,0)

END SUB

' ************************************************************

SUB LastKnownPos ()

   '-- get the coordinates of the last point in the line
   SET doc = Application.ActiveDocument
   SET comps = doc.ComponentSet
   SET comp = comps("Bomber-PTT: 52526")
   SET objSet = comp.ObjectSet

   SET obj = objSet(0)
   IF obj.Type = ObjectLine THEN
   SET branch = obj.Geom.BranchSet.Item(0)
   SET pts = branch.PointSet
   END IF
   '-- Set up a coordinate system and create a new drawing
   SET CS = Application.NewCoordinateSystem("Latitude / Longitude")
   SET nDwg = Application.ActiveDocument.NewDrawing("End-Bomber", CS)

   SET pointset = Application.NewPointSet
```

```
   SET point = Application.NewPoint

   '-- Edit: use pts.Count method to set the x & y coords for first point on line
   point.X = CDbl(pts(pts.Count-1).x) '--CDbl(record.data("SGT-Longitude"))
   point.Y = CDbl(pts(pts.Count-1).y) '--CDbl(record.data("Latitude"))
   pointset.Add point

   '-- Create a new geometric entity
   SET geom = Application.NewGeom(GeomPoint, PointSet)

   '-- Create new point object
   nDwg.ObjectSet.Add geom
   SET map = Application.ActiveDocument.ComponentSet("Map")
   SET nLayer = Application.ActiveDocument.NewLayer(nDwg)

   map.LayerSet.Add nLayer

   '-- assign a custom colour to the tagging operation for Bomber
   nDwg.PointBackground.DefaultValue.Formatting=application.NewColor("Purple",
255,0,255)
   nDwg.PointForeground.DefaultValue.Formatting=application.NewColor("Purple",
255,0,255)

END SUB

' ************************************************************
```

Map Components

Find a Map Window Extents, Center and Scale

http://69.17.46.171/Site/Thread.aspx?id=18137&ti=632736472139800000

artlembo at 1/23/2006 9:36 AM (#18138)

I can get the center and scale of a map window using:

```
Set MapWin = Application.ActiveWindow
Set WinPt = MapWin.Location
theScale = MapWin.Scale
```

However, if I don't know the width and height of the window, I can't program the actual geographic extents of the window. Does anyone know if it is possible to grab the extents of the current map window?

KlausDE at 1/23/2006 10:06 AM (#18139)

No, but the map window always is smaller than the ApplicationWindow and this object has Width and Height. So you can be sure that the following code will cover (slightly) more than the visible extend of the map window.

```
--VBScript
dx = Application.Width * MapWin.ScaleInternal / 2.0
dy = Application.Height * MapWin.ScaleInternal / 2.0

leftX = WinPt.X - dx
rightX = WinPt.X + dx
topY = WinPt Y + dy
bottomY = WinPt.Y - dy
```

Select Touch All Objects in All Map Layers that Touch a Bounding Object

http://69.17.46.171/Site/Thread.aspx?id=22016&ti=632847696209100000

sitesatlas at 6/1/2006 1:36 PM (#22024)

I want to take all the layers in my Map (Ports, Stations, Railroads, etc.) and select those parts that are touching the rectangle in the Bounding Box layer. I suppose I could add those two lines for all the 30 layers in my map, but I thought there would be an easier way. Couldn't I use a LayerSet operation or a loop or something here to automatically process all the map layers in the same way?

artlembo at 6/1/2006 1:51 PM (#22025)

right. Thats why I mentioned a loop. Remember, you have a layerSet object.

Do this:

```
Sub Main
  Set ZActiveDoc = Application.ActiveDocument
  Set ZCompSet = ZActiveDoc.ComponentSet
  Set ZProvinces = ZCompSet.Item("Provinces")
  Set ZBoundBox = ZCompSet.Item("BoundingBox")
  Set ZNewDraw = ZCompSet.Item("NewDraw")
  Set ZAnalyzer = ZActiveDoc.NewAnalyzer
  For each Layer1 in ZCompSet.Item("Map").LayerSet
    ZAnalyzer.SelectTouching layer1.Component, layer1.Component.ObjectSet, _
         ZBoundBox.ObjectSet
    Next
End Sub
```

sitesatlas at 6/1/2006 2:40 PM (#22026)

Thank you so much. I just made a couple little changes and it worked like a charm:

```
Sub Main
Set ZActiveDoc = Application.ActiveDocument
Set ZCompSet = ZActiveDoc.ComponentSet
Set ZBoundBox = ZCompSet.Item("Bounding Box")
Set ZAnalyzer = ZActiveDoc.NewAnalyzer
for each layer1 in ZCompSet.Item("Map").LayerSet
ZAnalyzer.SelectTouching layer1.Component, layer1.Component.ObjectSet,
ZBoundBox.ObjectSet
next
End Sub
```

Thematically Mapping Precipitation Amounts using Point Themes

http://69.17.46.171/Site/Thread.aspx?id=22795&ti=632901092226900000

KASSPER at 6/16/2006 8:29 PM (#22796)

I found where the National Weather Service stores the rainfall totals. I can write basic queries to present the data in thematic view, but I lack the query writing skills to make them look 100%. Would anyone like to write a few queries for a thematic rainfall display?

Here is where the NWS stores the live reports it is in SHP format and the recordings are in a gridded format: ftp://63.77.98.88/pub/rfcshare/precip_new/

I just want to say, great job with 7.0 and thank you in advance for any help.

adamw at 6/19/2006 6:24 AM (#22872)

How about this:

Unpack TAR.GZ into TAR. Unpack TAR into SHP / SHX / DBF / etc. Import SHP using File - Import - Drawing, SHP, using default import options. Open the imported drawing.

Click the Point Foreground button in the Format toolbar and select Theme. Set Field to Globvalue. Set Method to Equal Count. Set Align to -2 to avoid excessive rounding. Set Breaks

to 6. Click Tally. Set Palette to Precipitation. Press the Apply button in the toolbar right above the value list. Click OK.

Do the same for point background color. There are two alternatives:

Alternative 1: Transfer formatting from point foreground color: Click the Point Foreground color button in the Format toolbar and select Theme. Click the Save to File toolbar button. Edit the filename and click Save. Close the Format dialog. Click the Point Background color button in the Format toolbar and select Theme. Set Field to Globvalue. Click the Load from File toolbar button. Locate the file and click OK. Click the Lighten button in the toolbar (optional). Click OK.

Alternative 2: Set point style to the one that does not use the background color (eg, the "filled dot" style in the second position of the first row in the style drop down well).

rheitzman at 6/21/2006 2:50 PM (#23054)

Something to watch for using this method is that the resulting Equal Count distribution is llimited to the data in the sample which may not reflect a typical range of data. Equal Interval may be a better stat, but it still has data set related issues. One way to deal with this is to make a Theme.xml that is close then hand tweak it to cover the exepcted values. (The theme from either Interval type is identical.) For example:

The original began with:
<?xml version="1.0" encoding="UTF-8"?>
I 'think' that <xml> is required

```
<xml>
  <theme>
    <type>color[/type>
    <interval>true[/interval>
    <column>Globvalue[/column>
    <colors>
      [color>#f5e60c[/color>
      [color>#65e13d[/color>
      [color>#00b23d[/color>
      [color>#067d1d[/color>
      [color>#1f4791[/color>
    </colors>
    <colorMin>#cd853f[/colorMin>
    <colorMax>#1f4791[/colorMax>
    <colorDef>#cd853f[/colorDef>
    <values>
      [value>0.00[/value>
      [value>1.00[/value>
      [value>2.00[/value>
      [value>3.00[/value>
      [value>4.00[/value>
    </values>
  </theme>
</xml>
```

You can copy and past color/value pairs to cover the expected values. (I changed the < to [to avoid browser issues.) On the first pass don't worry about color, just get an equal number of color and value tags. Then load the theme to tweak the colors and save it for re-use.
==========
Points - note you can use square point symbols and vary the size to get a fairly decent looking gridded surface. Each zoom level requires a new size so that's a bit of a pain. Separate themes with varing point/zoom levels can help.

Another method is to build a static reusable cell layer (make one from the View, Grid/Graticule dialogs and use Spatial Overlays to move the data to the cells from the points.

Perfomance is an issue with either method.

Someday Manifold will get around to gridded data - if you ask for it.

Listing the Projection for Each Map Layer

```
' ************************************************************
' *     THE FOLLOWING MANIFOLD-L THREAD GIVES THE BACKGROUND.    *
' *          THIS IS ADAM'S SUGGESTION IN A SCRIPTED FORM.       *
' *                   L. Ketch - April 9, 2006                   *
' ************************************************************
'
' Sent:    Thu 04/06/2006 5:19 PM
' From:    David Weinschrott [david@prodevassociates.com]
' To:      manifold-l@lists.directionsmag.com
' Subject: [Manifold-l] Finding the bad apple
'
' I have a map I have been adding stuff to for a couple of weeks. All of a
' sudden things have slowed down a lot. My guess is that one of those layers
' has an odd projection.  In my understand, In order to find the bad apple I
' have to find each of those layers in the project mnanager and check its
' projection. Is there any other quicker way -- it would be cool to be able
' to put up a table that would list projections of each layer.  One could have
' a button that would change all layers to a selected projection, but that is
' a bit drastic when you don't know which one might need specific projection
' attention.
'
' David J. Weinschrott, PhD
'
' ------------------------------------------------------------------------
'
' Sent:    Fri 04/07/2006 10:06 AM
' From:    adamw@manifold.net
' To:      manifold-l@lists.directionsmag.com
' Subject: RE: [Manifold-l] Finding the bad apple
'
' > ... it would be cool to be able to put up a table that
' > would list projections of each layer.
'
```

```
' Here is what you could do:
'
' Create a table named "Data" with a text column named "Name". Open the table
' and add a record for each component you are interested in. Create a new query
' and set its text to:
'
'  SELECT [Name], CoordSys([Name] AS COMPONENT) [CoordSys] INTO [CoordSystems] FROM
[Data];
'
' Run the query. Open the resulting table. Right click the "CoordSys" column,
' select Format, set formatting style to "XML" and click OK. Resize the column.
' You should see the name of the coordinate system preset in the first 50 or
' so characters of the XML (between <name> and </name>).
'
' --
' Adam Wachowski
' Manifold Development Team
' ************************************************************

Option Explicit
```

Lorne on 5/8/2006 4:20 AM (#20896)
http://69.17.46.171/Site/Thread.aspx?id=20894&ti=632918264627670000

```
Sub Main()
  '   L. Ketch - April 9, 2006
  Dim Document
  Dim Component
  Dim CoordSys_Table
  Dim Column, Columns
  Dim Record, RecordSet
  Dim ActiveWin
  Dim CoordSys, CoordSys_ParmSet
  Dim MapLayer, MapLayers

  Set Document = Application.ActiveDocument
  Set ActiveWin = Application.WindowSet.ActiveWindow

  If ActiveWin.Component.Type <> ComponentMap Then
      Application.MessageBox "The Active Window must be a Map"
      Exit Sub
  End If
  Set MapLayers = ActiveWin.Component.LayerSet

  Delete_Component("Layer_CoordSystems")
  Set CoordSys_Table = Document.NewTable("Layer_CoordSystems")
  Set Columns = CoordSys_Table.ColumnSet

  Set Column  = Columns.Item(0)
  Column.Name = "MapLayer"
  Column.Type = ColumnTypeWText
  Column.Size = 100

  Set Column  = Columns.NewColumn
  Column.Name = "CoordSys_Name"
  Column.Type = ColumnTypeWText
```

```
Column.Size = 100
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "Datum"
Column.Type = ColumnTypeWText
Column.Size = 100
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "EllipsoidName"
Column.Type = ColumnTypeWText
Column.Size = 100
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "CoordSys_Unit"
Column.Type = ColumnTypeWText
Column.Size = 50
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "localOffsetX"
Column.Type = ColumnTypeFloat32
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "localOffsetY"
Column.Type = ColumnTypeFloat32
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "localScaleX"
Column.Type = ColumnTypeFloat64
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "localScaleY"
Column.Type = ColumnTypeFloat64
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "ScaleCorrectionX"
Column.Type = ColumnTypeFloat64
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "ScaleCorrectionY"
Column.Type = ColumnTypeFloat64
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "FalseEasting"
Column.Type = ColumnTypeFloat64
Columns.Add(Column)

Set Column  = Columns.NewColumn
```

```
Column.Name = "FalseNorthing"
Column.Type = ColumnTypeFloat64
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "majorAxis"
Column.Type = ColumnTypeFloat64
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "Eccentricity"
Column.Type = ColumnTypeFloat64
Columns.Add(Column)

Set Column  = Columns.NewColumn
Column.Name = "CoordSys_XML"
Column.Type = ColumnTypeWText
Column.Size = 2000  ' >1000 required here or the script fails
Columns.Add(Column)

Set RecordSet = CoordSys_Table.RecordSet

For Each MapLayer in MapLayers
   Set Component = MapLayer.Component
   Set CoordSys = Component.CoordinateSystem
   Set CoordSys_ParmSet = CoordSys.ParameterSet

   RecordSet.AddNew()
   Set Record = RecordSet.LastAdded

   Record.Data("MapLayer") =  Component.Name

   Record.Data("CoordSys_Name") =  CoordSys.Name
   Record.Data("Datum")         =  CoordSys.Datum.Name
   Record.Data("EllipsoidName") =  CoordSys.Datum.Ellipsoid.Name
   Record.Data("CoordSys_Unit") =  CoordSys.Unit.Name

   Record.Data("localOffsetX")  =  CoordSys_ParmSet.Item("localOffsetX").Value
   Record.Data("localOffsetY")  =  CoordSys_ParmSet.Item("localOffsetY").Value

   Record.Data("localScaleX")  =  CoordSys_ParmSet.Item("localScaleX").Value
   Record.Data("localScaleY")  =  CoordSys_ParmSet.Item("localScaleY").Value

   Record.Data("ScaleCorrectionX")  =  CoordSys_ParmSet.Item("ScaleX").Value
   Record.Data("ScaleCorrectionY")  =  CoordSys_ParmSet.Item("ScaleY").Value

   Record.Data("FalseEasting")  =  CoordSys_ParmSet.Item("FalseEasting").Value
   Record.Data("FalseNorthing")  =  CoordSys_ParmSet.Item("FalseNorthing").Value

   Record.Data("majorAxis")  =  CoordSys_ParmSet.Item("majorAxis").Value
   Record.Data("Eccentricity")  =  CoordSys_ParmSet.Item("Eccentricity").Value

   Record.Data("CoordSys_XML")  =  CoordSys.ToXML()
Next

CoordSys_Table.Open()
```

```
End Sub

' ************************************************************

Private Sub Delete_Component(ComponentName)
   Dim Component_Index
   Dim Components

   Set Components = Application.ActiveDocument.ComponentSet

   Component_Index = Components.ItemByName(ComponentName)
   If Component_Index < 0 Then Exit Sub

   Components.Remove(Component_Index)

End Sub

' ************************************************************
```

Checking the WindowSet For a MapComponent Then finding the Active Drawing

http://www.georeference.org/Forums/tabid/71/view/topic/forumid/4/postid/5441/Default.aspx
also see:
http://www.georeference.org/Forums/tabid/71/forumid/7/postid/174/view/topic/Default.aspx
which is NOT reproduced below.

gxdata
11/16/2004 7:28 PM

```
Sub Main

  Set windows = Application.WindowSet
  If windows.Count = 0 Then
    Application.Messagebox "No active component"
    Exit Sub
  End If ' -- fail if there are no opened windows

  Set window = windows.ActiveWindow
  Set theComponent = window.Component
  If theComponent.Type = ComponentMap Then ' drill down to active layer
     Set topDrawing = window.ActiveComponent
     Application.messagebox "Active Drawing is " & topDrawing.name, _
     "Map is " & theComponent.name

  Else
     Application.messagebox _
        "Activate the map (invalid component open)", _
        "Test for Map component: FAILED"
     Exit Sub
  End If
```

```
End Sub
```

Zoom to Selected Object in Map Window and Save an Image

Cannot retrieve the URL:
Probably Manifold©-L – Many June and July 2006 postings seem to be lost at:
http://lists.directionsmag.com/discussion/list.php?f=29
and
http://lists.directionsmag.com/discussion/search.php?f=29

```
Option Explicit

Sub Main
    ' mdsumner at 7/7/2006 4:37 PM (#24027)
    ' Here's a basic example in VBScript that just uses the
    ' geometric properties that might help you code the C# you want:
    '
    ' I added the Option Explicit, changed some of the variable
    ' names, changed the V6.5 code:
    '    Set doc = Application.ActiveDocument
    '    Set comps = doc.ComponentSet
    ' to V7 format
    '    Set Comps = Document.ComponentSet

  Dim Comps, Comp, MyMap, Obj, bBox, n

  Set Comps = Document.ComponentSet
    ' obtain drawing with area objects
  Set Comp = Comps("World_LoRes Drawing")

    '  obtain map to render from
  Set MyMap = Comps("Map")

    ' pick an object
  n = 0
  Set obj = comp.ObjectSet.Item(n)

    ' select it
  Obj.Selected = TRUE

    ' grab its bounding box "Rect" property
  Set bBox = obj.Geom.Box

    ' render to an image
  MyMap.RenderAreaTo "Image", 100, 100, bbox, TRUE

    ' optionally update the map window
    ' make sure it's the active window
  MyMap.Open
  Application.WindowSet.ActiveWindow.ZoomTo Obj

End Sub
```

Setting Map Layer Order by code

http://forum.manifold.net/Site/Thread.aspx?id=29866&ti=632984893201530000

KlausDE on 11/7/2006 9:34 AM (#29877)

```
mapLayerSet.Add newLayer, newLayerPosition
```

The second parameter is not documented in help but working perfectly well and visible in the object browser of VS or VBA.

Turning Layers On and Off via a Script

http://www.georeference.org/Forums/tabid/71/forumid/7/tpage/1/view/topic/postid/2101/Default.aspx#2105

mdSumner

04/27/2004 8:24 PM

You can obtain a specific layer by LayerSet.ItemByName

```
Sub Main
   Set Comps = Document.ComponentSet
   Set Comp = Comps("Map")
   For Each Layer in Comp.LayerSet
      Layer.Visible = FALSE

        ' Application.MessageBox Layer.Component.Name & " is turned off"
   Next
End Sub
```

Collect a Map Window Parameter Set (CoordSys, Width, CenterX, etc)

```
Sub Main

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

'  Set WinSet = Application.WindowSet
'  Set MapWin = WinSet.Item("WT140245_Gordon_Map")

  Set Comps = Document.ComponentSet
  'Set Map = Comps.Item("WT140245_Gordon_Map")
  Set Map = Comps.Item("WT140253_Helene_Map")

  Set CoordSys = Map.CoordinateSystem
```

```
  Map.Open
  Set MapWin = Application.ActiveWindow
  Set MapWin_Rect = MapWin.Bounds

'  Set Pt = Application.NewPoint(MapWin_Rect.Xmin, MapWin_Rect.Ymin)
'  Set NewGeom = Application.NewGeom(GeomPoint)

  Application.MessageBox _
       "Property" & VbTab & VbTab & "Value" & VbCrLf & _
       "Coordinate Sys:"  & VbTab & CoordSys.Name & VbCrLf & _
       "MinX:" & VbTab & VbTab & MapWin_Rect.Xmin & VbCrLf & _
       "MaxX:" & VbTab & VbTab & MapWin_Rect.Xmax & VbCrLf & _
       "MinY:" & VbTab & VbTab & MapWin_Rect.Ymin & VbCrLf & _
       "MaxY:" & VbTab & VbTab & MapWin_Rect.Ymax & VbCrLf & _
       "Width:" & VbTab & VbTab & MapWin_Rect.Width & VbCrLf & _
       "Height:" & VbTab & VbTab & MapWin_Rect.Height & VbCrLf & _
       "CentreX:" & VbTab & VbTab & MapWin_Rect.Center.X & VbCrLf & _
       "CentreY:" & VbTab & VbTab & MapWin_Rect.Center.Y

End Sub

' ************************************************************************
```

Miscellaneous

Manifold Should NEVER Crash

http://forum.manifold.net/Site/Thread.aspx?id=22237&ti=632852579889430000

probio at 6/6/2006 8:48 PM (#22238)
I have got a script that imports a layer from a number of project files .map (about 10) and after importing will combined all the layers into one layer. The script is ran from my local drive while the project files are stored in a server. Half way through the process, manifold crashes resulted in a Windows error report dialogue appearing. My question is whether this is a bug or some thing else. As i recall i do not have any problem with this when i upgrade from 6.0 to 6.50. Thanks

adamw at 6/7/2006 6:19 AM (#22261)
This is a bug. **Running a script should never crash**. I suggest you send a report to tech support and offer to upload example MAP files that can be used to reproduce the problem.

http://forum.manifold.net/Site/Thread.aspx?id=31334&ti=633008135566530000

adamw on 12/2/2006 6:20 AM (#31365)
No, that's not normal. It is expected that some data sets will be too large to handle, but **Manifold should not crash, ever. Anything that results in a crash is a bug.**

I suggest you take this to tech support.

You Can Not Currently Contour Data Using a Script

http://lists.directionsmag.com/discussion/read.php?f=29&i=41189&t=41187
Adam Wachowski

Calculating Spherical Distances and Areas in Manifold© — (HURDAT)

My Subroutine solutions for this (*L. Ketch*):
Active column and Form based routines follow.
Note that this can be done much easier with V7 SQL using either the *DistanceEarth* or *LengthEarth* functions.

```
Option Explicit

 ' ------------------------------------------------------
 ' UPDATING THE "SystemSpeed_kts" COLUMN IN THE HURDAT TABLE
 ' ------------------------------------------------------
```

```
 ' The Length and Area SQL Keywords will only return
 ' values in the projected space.  There appears to
 ' no easy SQL approach to returning these values
 ' using spherical algebra (No longer true :-) as of V7).

 ' Working directly with the Object set provides
 ' a way to get to ellipsoidal Length calculations.
 ' Unfortunately, it is slow.  One area slowing
 ' the processing is the updates to the status bar
 ' and progress bar.  Don't do this for every pass
 ' through the loop... once every 100 passes is fine.

 ' The process below can be replicated using
 ' Active Columns.  Example.

 ' THE ACTIVE COLUMN FUNCTION BELOW CALCULATES THE
 ' SAME VALUES THAT MAPINFO© WOULD GIVE.
 '
 ' Function Spherical_Speed
 '      ' Table is Azimuthal Equidistant World Geodetic 1984 (WGS84)
 '    Set objs = Table.Owner.ObjectSet
 '    Set obj =  objs(objs.ItemByID(Record.Data("ID")))
         ' convert metres to Nautical Miles
 '    Length_nm = obj.Geom.Length * 0.000539594075
 '    Spherical_Speed =  Round(Length_nm / 6.0, 1)
 ' End Function

 ' INVESTIGATE THE "DISTANCEEARTH" SQL KEYWORD.
 ' DistanceEarth(geomA, geomB[, unit])
 ' Returns ellipsoidal distance between given points.

 ' NOTES:
 ' 1.  HURDAT DOES NOT HAVE TO BE PROJECTED... THIS
 '     ROUTINE WORKS ON LATITUDE/LONGITUDE DRAWINGS
 '     AS WELL AS PROJECTED DRAWINGS.
 ' 2.  THE CALCULATED SPEEDS ARE WITHIN 0.1 KNOTS
 '     OF THE VALUES DERIVED FROM Mapinfo© - Mapinfo© appears
 '     to automatically use default ellipsoid calculations
 '     for unprojected drawings.

' ************************************************************

Sub Main
     Form.Visible = True
End Sub

' ************************************************************

Sub cmd_Proceed_Click()
   Update_SysSpd_via_Objects()
End Sub
```

```
' ************************************************************

Sub cmd_Close_Click()
   Form.Visible = False
End Sub

' ************************************************************

Sub Update_SysSpd_via_Objects()
   Dim Document, Components, Component, Component_Index
   Dim Owned_Table
   Dim Obj, Objs
   Dim Length_nm, Speed
   Dim ObjID, RecordID
   Dim Record, Records, RecordCount

   Set Document = Application.ActiveDocument
   Set Components = Document.ComponentSet
   Component_Index = Components.ItemByName("HURDAT_2004 Drawing")
   Set Component = Components(Component_Index)
    ' OR,
    ' Set Component =
Components(Components.ItemByName("HURDAT_Projected Drawing"))

   Set Owned_Table = Component.OwnedTable
   Set Records = Owned_Table.RecordSet
   RecordCount = Records.Count

   Set Objs = Component.ObjectSet

   Prog_Progress.Min = 0
   Prog_Progress.Max = RecordCount

   SBar_ProgressReport.Panels(2).Text = cStr(RecordCount) & " Records
in HRUDAT"

   For Each Obj in Objs

        ' Metres to Nautical Miles conversion factor: 0.000539594075
      Length_nm = Obj.Geom.Length * 0.000539594075
      Speed =  Round(Length_nm / 6.0, 1)
      OBjId = Obj.ID

      RecordID = Records.ItemByID(ObjID)
      Set Record = Records.Item(RecordID)
      Record.Data("SysSpd_Manifold") = Speed

      If  RecordID Mod 100 = 0 Then
            ' DO NOT update the bars too often. Doing so will
            ' bog down code exection.
         SBar_ProgressReport.Panels(1).Text = cStr(RecordID)
```

```
         Prog_Progress.Value = RecordID
      End IF

   Next

   Form.Visible = False
End Sub

' ************************************************************

Sub Update_SysSpd_via_Records()
   Dim Document, Components, Component
   Dim Owned_Table
   Dim Obj, Objs
   Dim Length_nm, Speed
   Dim Record, Records, TotalRecords, RecordCounter

   Set Document = Application.ActiveDocument
   Set Components = Document.ComponentSet
   Set Component = Components(Components.ItemByName("HURDAT_2004
Drawing"))

   Set Owned_Table = Component.OwnedTable
   Set Records = Owned_Table.RecordSet
   Set Objs = Component.ObjectSet

   TotalRecords = Records.Count

   Prog_Progress.Min = 0
   Prog_Progress.Max = TotalRecords - 1

   SBar_ProgressReport.Panels(2).Text = cStr(TotalRecords) & " Records
in HRUDAT"

   For RecordCounter = 0 to TotalRecords - 1
      Set Record = Records(RecordCounter)
      Set Obj =  Objs(Objs.ItemByID(Record.Data("ID")))
        ' OR
        ' Set Obj = Objs.Item(RecordCounter)

        ' Metres to Nautical Miles conversion factor: 0.000539594075
      Length_nm = Obj.Geom.Length * 0.000539594075
      Speed =  Round(Length_nm / 6.0, 1)

      Record.Data("SysSpd_Manifold") = Speed

      If  (RecordCounter + 1) Mod 100 = 0 Then
            ' DO NOT update the bars too often. Doing so will
            ' bog down code exection.
         SBar_ProgressReport.Panels(1).Text = cStr(RecordCounter + 1)
         Prog_Progress.Value = RecordCounter
      End IF
```

```
   Next

   Form.Visible = False

End Sub

' ************************************************************
```

Create a Component at the Root Level (Not in an Existing Folder)

http://69.17.46.171/Site/Thread.aspx?id=18274&ti=632738929572000000

p1000 at 1/25/2006 5:08 PM (#18275)

I am trying to generate a new Table object in a script using the command:

```
table = doc.NewTable("MEMORY", columnSet, false);
```

I am experiencing the following problem. My script is in a "Scripts" folder. When the script is run it creates the Table in the "Scripts" folder as well. I would like the Table to be generated in the root of the document.

I can't figure out from the reference section of the user manual how this can done. Is there a focus which determines where the new component is put? How can the focus be moved around?

Thanks,

P1000

mdsumner at 1/26/2006 5:22 PM (#18325)

Here's a workaround (in VBScript):

```
Sub Main
   Set doc = Application.ActiveDocument
   Set table = doc.NewTable("MEMORY", , False)
   Table.Folder = Nothing
```

Copy a Component from another MAP File

http://69.17.46.171/Site/Thread.aspx?id=1048&ti=632109909000000000

adamw on 1/29/2004 1:35 PM (#1167)

> *mdsumner - 2004-01-27 3:56 PM - ... formatting is not preserved for me (when I do it in a script - it's fine with GUI copy/paste), either from a saved document or the active document.*

That's because what you do in GUI is probably not what you do in a script, in that in GUI you paste to the Project pane but in script you paste to the drawing window. :-)

Try pasting with Document.Paste:

```
Sub Main
    ' copy component from foreign project
    Set doc = Application.NewDocument("C:/Temp/Test.map", true)
    Set comps = doc.ComponentSet
    Set comp = comps("Drawing")
    comp.Copy False ' copy component

    ' paste component to active project

    Set thisDoc = Application.ActiveDocument
    thisDoc.Paste      ' paste component
End Sub
```

Applying Thematic Legends

http://69.17.46.171/Site/Thread.aspx?id=18864&ti=632751756750600000

njengler at 2/10/2006 1:41 PM (#18938)

In order to apply thematic legends, in your case Equal Count, colouring, you must use SetUniqueValues() or one of the other thematic settings. This will give you a range of foreground, background, style, and size values which you can then access with Formatting() and set individually. See code below:

```
mColumn = "some column object"
mDrawing.LineForeground.SetUniqueValues(mColumn)
For i = 0 To mDrawing.LineForeground.Values.Count - 1
   mDrawing.LineForeground.Values(i).Formatting = gApplication.NewColor("x",
'R','G','B')
Next
mDrawing.Refresh()
```

Calling One Script from another Script

Date: Thu 02/16/2006 3:35 AM

From: Adam Wachowski [adamw@manifold.net]

To: Manifold List

Subject: Re: [Manifold-l] May I know how to call a script in ASP code?

http://lists.directionsmag.com/discussion/read.php?f=29&i=41563&t=41548

> May I know how to call a script in ASP code?

Similarly to calling a script from another script:

```
var scr = mapserver.Document.ComponentSet("Script");
scr.Run();
```

The called script should not reference the ActiveDocument property of the Application object, and should instead locate the document in the DocumentSet collection of the same object.

A .NET script can also obtain the document using the Document property of the Context object.

Adam Wachowski

Proper way to Assign an Object to a Component Added by the .NEW Method

From Adam's comment in the "Bound Labels Created by Script do Not Display" topic, http://69.17.46.171/Site/Thread.aspx?id=22339&ti=632854363953230000
one should assign an object variable to a new component when it is created because you can not rely on the newly added labels component to be the last one in the component set.

DO THIS:

```
Set Labels_E = Document.NewLabels(Temp_Str & "_Labels_E", Points, TRUE)
```

NOT THIS:

```
Document.NewLabels Temp_Str & "_Labels_E", Points, TRUE
Set Labels_E = Comps.Item(Comps.Count - 1)
```

Adding a DLL Reference to a VBScript

http://69.17.46.171/Site/Thread.aspx?id=20106&ti=632798794218230000

TimOsborn at 4/3/2006 6:22 PM (#20107)

I understood I could invoke a form that was coded in a .NET DLL. This works fine in VB.NET for example:

```
Dim S As New PI.frmPiScoreboard
S.Visible = True
```

But the same thing in a VB.NET script in Manifold raises:

```
"Exception has been thrown by the Target of an Invocation"
```

Can anyone help?

Thanks,

Tim

TimOsborn at 4/4/2006 8:34 PM (#20168)
Fixed it! -- so I'm going to reply to my own post.

The DLL has to be in Program Files\Manifold.System. Anywhere else and it compiles but won't run. How whacky is that. I can't find any reference to this in the documentation.

Tim

spoedniek at 4/5/2006 2:46 AM (#20183)
Does it not work when you add the reference using Script/References? Browse to the DLL and add it to the list.

Henry

TimOsborn at 4/5/2006 4:14 PM (#20208)
Hi Henry,

No, that's the point. I'd certainly added the reference to the DLL, but it only works if the DLL is in Manifold.System.

(It wouldn't compile if the reference was missing)

Tim

Implementing "Dissolve"

http://69.17.46.171/Site/Thread.aspx?id=18945&ti=632753461463730000

ky Schemer at 2/11/2006 10:21 AM (#18946)
Greetings,

I am trying to implement a "dissolve" function in C# so that I can automate the processing of map objects in numerous maps (doing dissolve by hand gets tedious, fast). From what I can tell based on comments here and my own experiences, the correct approach appears to be to select the desired objects in the drawing, get the GeomSet from Drawing.Selection.GeomSet and then apply the Union method to return the combined geometry.

When I do this, however, none of my column data for the "old" geometry transfers to the new object. I get a new, unified object, but its data fields in the drawing table are all set to their

default values. I do have transfer rules set to Copy/Copy. "Manually" applying the Union transfrom in the toolbar does the right thing. The script, however, loses the data.

Here's a C# snippet:

```
    ' C#
Application app= Context.Application;
Document doc= (Document) app.ActiveDocument;
ComponentSet cset= doc.ComponentSet;
ObjectSet oset;

Drawing dwg= (Drawing) cset["Messy Drawing"];
oset= dwg.Selection;
if ( oset.Count > 0 ) {
 GeomSet gset= oset.GeomSet;
 Geom geo= gset.Union(dwg.Epsilon);
 gset= dwg.Selection.GeomSet;
 dwg.Clear(true);
 oset= dwg.ObjectSet;
 if ( geo != null ) oset.Add(geo);
}
```

Am I doing something wrong? Or is there more to this than I think?

-SS

sky schemer at 2/12/2006 1:02 PM (#18952)

Note that I was able to get this working. I Just had to copy the column data myself in the script. :(

Use "Union Rather then "Dissolve" in VB Script

http://69.17.46.171/Site/Thread.aspx?id=20164&ti=632804672072100000

- There is no Script Dissolve, but you can use Union.
- Dissolve is basically the same as a Union based on a specific table.

dgillgren on 4/4/2006 6:53 PM (#20165)

Is there any way call functions such as "Dissolve" in a VB Script, or will I need to create my own?

Darryl

mechalas on 4/5/2006 7:19 AM (#20186)

See this thread on the archived forum site.
http://www.georeference.org/Forums/tabid/71/forumid/7/tpage/1/view/topic/postid/13583/Default.aspx#13586

dgillgren on 4/11/2006 7:12 PM (#20408)

Well worked out my dissolve function, Thanks for those hints. I found some useful stuff in the archives as well.

Now my problem is that when I come to select the polygons I wish to dissolve the only way I can do it is by opening the table, which means the table pops up on the screen. Is there a way to suppress this and still select the polygons?

What am I missing?

Rest of Program goes here ...

```
Application.StatusText = "Adding Records to MAStatus Drawing"
For i = 0 to MACodeRecSet.Count -1
   theMA = MACodeRecSet.Item(i).DataText("MACode")

   Set MAQuery = Components(Components.ItemByName("MAQuery"))

   ' Note:
   ' To Run this Query the "ANSI compatible" check box has to
   ' be ticked in the MAQuery properties

     ' Original line by dgillgren at 4/11/2006 7:12 PM (#20408) below:
   'theText = "SELECT * FROM [" & blockDrawing.Name & "] & _
   '      WHERE [MACode] = '" & theMA & "';"

     ' Adam suggested changing it to an Update query t:
   'theText = "UPDATE [" & blockDrawing.Name & "] & _
   '      SET [Selection (I)] = ([MACode] = '" & theMA & "');"

   MAQuery.Text = theText
   MAQuery.Run

   ' The Table.Open not required if Adam's suggestion below is used.
   ' Note:
   ' No records selected unless table is Open - displays each selection onscreen
   ' MAQuery.Table.Open()

   Set selBlocks = blockDrawing.Selection

   '  Add dissolved polygons to separate drawing
   maObjects.Add selBlocks.GeomSet.Union
   Set theRecord = maRecords(maRecords.Count-1)
   theRecord.Data("MACode") = theMA
Next
```

adamw at 4/12/2006 7:12 AM (#20415)

Rewrite the SELECT query as an UPDATE query:

```
'VBScript
theText = "UPDATE [" & blockDrawing.Name & "] SET [Selection (I)] = ([MACode]
= '" & theMA & "');"
```

Changing Drawing Area Colours and Themes

http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/3227/Default.aspx#3260

```
Sub Main
  Set doc = Application.ActiveDocument
  Set drw = doc.ComponentSet("Drawing")
  Set fmt = drw.AreaBackground

  ' set formatting
  fmt.Set Application.NewColor("", 0, 220, 0)

  ' set thematic formatting
  green = 220
  fmt.SetUniqueValues("Name")
  For Each val In fmt.Values
    val.Formatting = Application.NewColor("", 0, green, 0)
    green = green + 20
   Next

  ' set formatting using XML
  xml = ""

  fmt.LoadFrom xml
End Sub
```

If all you want is to copy it, use Format.SaveToFile and Format.LoadFromFile:

```
Sub Main
  Set doc = Application.ActiveDocument
  Set drw = doc.ComponentSet("Drawing")
  Set fmtSrc = drw.AreaBackground
  Set fmtTgt = drw.AreaForeground

  fmtSrc.SaveToFile "c:\fmt.xml"
  fmtTgt.SetUniqueValues "Name"
  fmtTgt.LoadFromFile "c:\fmt.xml"
End Sub
```

This also works:

```
Sub Main
   set theComponentSet = Application.ActiveDocument.ComponentSet
   set theDrawing = theComponentSet.Item("Ag Census Data")
   set datacolumn = theDrawing.OwnedTable.ColumnSet.Item("data")
   Application.Messagebox datacolumn.name
   set fmt = theDrawing.AreaBackground

   breaks = 10
   fmt.SetEqIntervals datacolumn, breaks
   theDrawing.Refresh
```

```
End Sub
```

Convert Table Query to Delimited Text Stored in Comments Component

Highly modified version of mdsumner and VON examples found at URL below
See: C:\Documents and Settings\Lorne_Limited\My Documents\Manifold\VBscript\ TableQuery_2_DelimitedText.map

http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/12575/Default.aspx#12599

```
Option Explicit

' Author: Lorne Ketch
'
' NOTE:
' Idea stems from MIKE SUMNER'S example found at the URL below.
' Many options added.
'
http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/12575/
Default.aspx#12599

' ************************************************************

Sub Main
  Dim Table_Name, Column_Names()

  Table_Name = "WxStns"
  Redim Column_Names(4)
  Column_Names(0) = "Prov_State_ID"
  Column_Names(1) = "ICAO"
  Column_Names(2) = "Syno_Num"
  Column_Names(3) = "Lat_Dec"
  Column_Names(4) = "Long_Dec"

    ' Chr(9) for TAB
  Create_Delimited_Table Table_Name, Column_Names, "fixed width", FALSE, TRUE
End Sub

' ************************************************************

Sub Create_Delimited_Table ( _
        ByRef Source_Table_Name, _
        ByRef Column_Names(), _
        ByRef Separator, _
        ByRef Quotes, _
        ByRef Include_ColumnNames )

     ' Author: Lorne Ketch
     '
     ' NOTE:
     ' Idea stems from MIKE SUMNER'S example found at the URL below.
     ' Many options added.
```

```
    '
http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/12575/
Default.aspx#12599
    '
    ' Source_Table_Name
    ' ~~~~~~~~~~~~~~~~
    ' The table (name) from which records will be extracted via a Query.
    '
    ' Column_Names()
    ' ~~~~~~~~~~~~~~
    ' An array that contains the column names to be used in the Query.
    '
    ' Separator
    ' ~~~~~~~~~
    ' Single character used to separate the column values in the
    ' formatted text written to comment component "Formatted Records".
    ' Examples of values passed to the function:
    '   " "             space separated values
    '   ","             CSV
    '   Chr(9)          tab delimited values
    '   "|"             bar separated values
    '   "Fixed Width"   All entries in a column have the same width.
    '                   Each column width equals:
    '                     longest entry in that column + 1 character
    '
    ' Quotes
    ' ~~~~~~
    ' TRUE = Adds double-quotes to the text column values.
    '   ColumnTypeAChar
    '   ColumnTypeAText
    '   ColumnTypeWChar
    '   ColumnTypeWText
    '
    ' Include_ColumnNames
    ' ~~~~~~~~~~~~~~~~~~~
    ' The first line will contain the column names. If Quotes = True,
    ' then double-quotes will surround the column names.
    '
    ' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    ' This function will SELECT the columns in "Column_Names()"
    ' FROM the table named "Source_Table_Name".  The resulting
    ' table display will be moved into a Comments component named
    ' "Formatted Records".
    '
    ' Using a TAB separator will allow direct copy and paste into a
    ' spreadsheet like Excel which will recognize the TABs and
    ' automatically parse the fields correctly to separate cells;
    ' however, you can use any separator character(s).
    '
    ' Text in "Formatted Records" is automatically copied to the
    ' Windows Clipboard at the end of the routine.
    ' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  Dim Comps, Qry
  Dim Report, Report_Comp
  Dim Rcrd, Rcrds
  Dim Col, Cols
```

```
Dim Col_Index, Last_Col_Index
Dim ColName_String
Dim Add_Quotes
Dim Fixed_Width
Dim Column_Widths(), Column_Width, W
Dim Str, Padding
Dim Record_Count

Set Comps = Document.ComponentSet

   ' Make sure the Query component exists
If Comps.ItemByName("Query") < 0 Then
   Set Qry = Document.NewQuery("Query")
Else
   Set Qry = Comps.Item("Query")
End If

   ' Does the Comment Component used for the
   ' Formatted Records exist?
If Comps.ItemByName("Formatted Records") < 0 Then
   Set Report_Comp = Document.NewComments("Formatted Records")
Else
   Set Report_Comp = Comps.Item("Formatted Records")
   Report_Comp.Text = ""
End If

Last_Col_Index = uBound(Column_Names)

   ' Build the Column name string that will be used in the Query
For Col_Index = 0 to Last_Col_Index
   If Col_Index < Last_Col_Index Then
      ColName_String = ColName_String & "[" & Column_Names(Col_Index) & "],"
   Else
      ColName_String = ColName_String & "[" & Column_Names(Col_Index) & "]"
   End If
Next

Qry.Text = "SELECT " & ColName_String & " FROM [" & Source_Table_Name & "];"
Qry.Run

Set Rcrds = Qry.Table.RecordSet
Set Cols  = Qry.Table.ColumnSet

   ' If fixed width columns were specified, then find the width
   ' required for each column - based upon the length of the
   ' longest entry in that column which might include the
   ' column name as well as the column data.  Also depends
   ' upon whether quotes are required around text fields.
If uCase(Separator) = "FIXED WIDTH" Then
    Fixed_Width = True
    ReDim Column_Widths(Last_Col_Index)

    For Col_Index = 0 to Last_Col_Index

       If Include_ColumnNames Then
          If Quotes Then
             Column_Width = Len(Column_Names(Col_Index)) + 3
```

```
            Else
               Column_Width = Len(Column_Names(Col_Index)) + 1
            End If
         Else
            Column_Width = 0
         End If

         Set Col = Cols(Col_Index)
         Add_Quotes = FALSE

         If Quotes AND ( Col.Type = ColumnTypeAChar OR _
                         Col.Type = ColumnTypeAText OR _
                         Col.Type = ColumnTypeWChar OR _
                         Col.Type = ColumnTypeWText  _
                       ) Then Add_Quotes = TRUE

         For Each Rcrd in Rcrds
            If Add_Quotes Then
                W = Len(cStr(Rcrd.Data(Column_Names(Col_Index)))) + 3
            Else
                W = Len(cStr(Rcrd.Data(Column_Names(Col_Index)))) + 1
            End If

            If W > Column_Width Then Column_Width = W
         Next

         Column_Widths(Col_Index) = Column_Width

    Next

Else
    Fixed_Width = False
End If

    ' Insert the column names into the first line if requested.
If Include_ColumnNames Then
  For Col_Index = 0 to Last_Col_Index
        If Quotes Then
           Str = """" & Column_Names(Col_Index) & """"
        Else
           Str = Column_Names(Col_Index)
        End If

        If Fixed_Width Then
             ' Padding = String( Column_Widths(Col_Index) - Len(Str), 32 )
            Padding = Space( Column_Widths(Col_Index) - Len(Str))
            Str = Str & Padding
            Report_Comp.AddText (Str)
        Else
            If Col_Index = Last_Col_Index Then
               Report_Comp.AddText (Str)
            Else
               Report_Comp.AddText (Str & Separator)
            End If
        End If
  Next
```

```
      Report_Comp.AddText VbCrLf
   End If

   Record_Count = Rcrds.Count

       ' Final Loop: Add the formatted data for each record
   For Each Rcrd in Rcrds

      Report = ""

      For Col_Index = 0 to Last_Col_Index

         Set Col = Cols(Col_Index)
         Add_Quotes = FALSE
         If Quotes AND ( Col.Type = ColumnTypeAChar OR _
                         Col.Type = ColumnTypeAText OR _
                         Col.Type = ColumnTypeWChar OR _
                         Col.Type = ColumnTypeWText  _
                       ) Then Add_Quotes = TRUE

         If Add_Quotes Then
            Str = """" & cStr(Rcrd.Data(Column_Names(Col_Index))) & """"
         Else
            Str = cStr(Rcrd.Data(Column_Names(Col_Index)))
         End If

         If Fixed_Width Then
              ' Padding = String(Column_Widths(Col_Index) - Len(Str), 32)
            Padding = Space(Column_Widths(Col_Index) - Len(Str))
            Str = Str & Padding
            Report = Report & Str
         Else
            If Col_Index = Last_Col_Index Then
               Report = Report & Str
            Else
               Report = Report & Str & Separator
            End If
         End If

      Next

      Report_Comp.AddText Report & VbCrLf

      Application.StatusText = cStr(Rcrd.Index + 1) & " of " & cStr(Record_Count)

   Next

   Report_Comp.Open

   Report_Comp.Copy  ' Send a copy to the Windows Clipboard

End Sub

' ************************************************************
```

Search All Scripts for a Specific String

http://www.georeference.org/Forums/tabid/71/forumid/4/tpage/1/view/topic/postid/10586/Default.aspx#10586

Paquet
06/28/2005 10:26 AM
Tommy
07/14/2005 5:50 PM

```
Sub Main
  dim app,doc,comps
  set app = application
  set doc=app.activedocument
  set comps=doc.componentset

  sToFind=app.inputbox("Enter string to search for","")

  for each comp in comps
     if comp.typename = "Script" then
        sText=comp.text
        lFound=InStr(1,sText, sToFind, 1)
        if lFound <> 0 AND not isnull(lFound) then
           comp.open
        end if
     end if
  next

End Sub
```

Set Default Point Size and Style

http://www.georeference.org/Forums/tabid/71/forumid/7/tpage/1/view/topic/postid/1245/Default.aspx#1246

mdSumner
02/12/2004 10:37 PM

You have to delve into the Format object of the drawing (which is read-only), into its FormatValue object, and change its Formatting property (which is read/write). This example might help you get started:

```
Sub Main
   Set Comps = Document.ComponentSet
   Set Comp = Comps("Drawing")
     ' increase the point size
   Set Format = Comp.PointSize
   Set DefaultValue = Format.DefaultValue

   defaultValue.Formatting = 40

     ' change the point style
   Set Format = Comp.PointStyle
   Set DefaultValue = Format.DefaultValue
```

```
   DefaultValue.Formatting = 2
End Sub
```

Query Caching – BE CAREFUL when rerunning the same Query.Text via Code.

Note to me…. L. Ketch

I was having a lot of problems with determining how many Records were selected in a drawing after upgrading to V7. Code like that below often gave incorrect results when repeatedly run (first time correct but subsequent runs wrong if the number of objects selected changed). It turns out that Query results are cached for greater speed but you may need to force a recalculation. You could do this by setting Query.Text="", then resetting the original query. It also turns out that Manifold© provides direct support for doing this as Adam explains below.

```
Query.Text = "Select * From [Drawing] where [Selection (I)];"
Set Table = Query.Table
Application.Messagebox cStr(Table.Recordset.Count) & " Records are Selected"
```

From: adamw@manifold.net
To: manifold-l@lists.directionsmag.com
Sent: Tue 05/23/2006 10:18 AM
Subject: RE: [Manifold-l] Problem: Selected Records Count
http://lists.directionsmag.com/discussion/read.php?f=29&i=42207&t=42207

```
> Open Drawing and select one or more objects. Run the
> script. The first time I run the script, the correct number of
> selected records is returned. Select a different number of objects in
> Drawing and run the script again. I am finding that I get the same
> number as the first run.
```

This happens because **the query caches the results of the previous execution**.

To force the query to recompute the results, use Query.RunEx:

```
 ...
 Query.Text = "Select * From [Drawing] where [Selection (I)];"
 Query.RunEx True
 Set Table = Query.Table
```

Adam Wachowski

Cannot Link a Query Programmatically

http://69.17.46.171/Site/Thread.aspx?id=22181&ti=632851747952570000

kuldeep78 at 1/12/2006 5:44 AM (#17830)

Hi,
In manifold application we can import a drawing from query of the same project. How can we do that programmatically? Kuldeep

adamw at 1/13/2006 7:08 AM (#17877)

There is no easy way to do this programmatically, although you obviously can create a script which would run the query, loop through the resulting records, and create a drawing object for each record.

willh at 6/6/2006 7:13 AM (#22201)

You cannot programmatically link a drawing (send in a request!); what you can do is create a new drawing and modify your SQL to be an INSERT and use that to populate the new drawing with the result data--this is MUCH faster than iterating through a loop.

Zoom to Selected Areas (with additional border)

http://69.17.46.171/Site/Thread.aspx?id=24023&ti=632881304717330000

mdsumner on 7/7/2006 4:37 PM (#24027)

Here's a basic example in VBScript that just uses the geometric properties that might help you code the C# you want:

> *I have changed Mike's original VBscript:*
>
> 1. *Uses the currently selected object or objects.*
> 2. *Checks to make sure that at least one object has been selected.*
> 3. *Allows you to zoom to the select objects bounding box with an additional margin around the box.*
>
> *L. Ketch*

```
' ****************************************************************

Option Explicit

Sub Main
  Dim Comp, Comps
  Dim Map, Objs, bBox, NewBox
  Dim n
```

```
  Set Comps = Document.ComponentSet

    ' obtain drawing with area objects you want to select.
  Set Comp = Comps("Northern_Hemisphere")

    ' obtain map to render from
  Set Map = Comps("Map")

    ' Grab the currently selected object(s) in Comp
  Set Objs = Comp.Selection
  If Objs.Count < 1 Then
     Application.Messagebox "You have not selected any objects in drawing """
& _
          Comp.Name & """"
     Exit Sub
  End If

    ' grab its bounding box "Rect" property
  Set bBox = Objs.GeomSet.Box
  Set NewBox = Resize_DrawingBox( bBox, 0.1, 0.1 )

    ' Resize bBox to allow allow additional space around the selected
object(s)

   ' render to an image
'  map.RenderAreaTo "Image", 100, 100, bbox, true

    ' optionally update the map window
  Map.Open  'make sure it's the active window
  Application.WindowSet.ActiveWindow.ZoomTo NewBox

End Sub

' ************************************************************

Private Function Resize_DrawingBox( _
           ByRef OldBox, _
           ByRef X_Percent, _
           ByRef Y_Percent)

     ' Receives a Rect Object and enlarges or reduces it by the
     ' given percentages.

     ' The function creates a Rect Object that is sized X% wider
     ' and Y% higher than the Original Rect.  You can use the new
     ' Rect in the "ZoomTo Rect" MapWindow Method to add some
     ' "white space" around the original box.  Just personal opinion but
     ' I think the maps look better zoomed out a bit.
     '
     ' NOTE: In many cases, the Y extend does not seem to
     '       affect how Manifold actually zooms a MapWindow.
     '       X dominates while the changed Y sometimes seems
     '       to be ignored in the final map widow display.
     '       This seems related to where the geographic
     '       objects actually reside on the final map
     '       but I don't have a good handle on this yet.
```

```
      '       One could use a single % change for both
      '       width and height and often end result would
      '       be similar.

      ' Example Useage:
      ' ~~~~~~~~~~~~~~
      ' We want to ZoomTo Component "Lines" but allow a
      ' buffer around the original Lines rectangle.
      '   Set Win_Boundaries = Resize_DrawingBox(Lines.ObjectSet.GeomSet.Box,
0.10, 0.05)
      ' Win_Boundaries is now a Rect Object sized 10%
      ' larger in the X direction and 5% larger in the Y
      ' direction than the original Lines bounding box.
      '   Map_Window.ZoomTo(Win_Boundaries)

   Dim X_Max, X_Min, Y_Max, Y_Min
   Dim X_Change, Y_Change

   Set Resize_DrawingBox = OldBox

     ' The RectOobject's min and max properties return
     ' values that are in the drawing's projection.

   X_Max = Resize_DrawingBox.XMax
   X_Min = Resize_DrawingBox.XMin
   Y_Max = Resize_DrawingBox.YMax
   Y_Min = Resize_DrawingBox.YMin

   X_Change = (X_Max - X_Min) * X_Percent / 2
   Y_Change = (Y_Max - Y_Min) * Y_Percent / 2

   Resize_DrawingBox.XMin = X_Min - X_Change
   Resize_DrawingBox.XMax = X_Max + X_Change

   Resize_DrawingBox.YMin = Y_Min - Y_Change
   Resize_DrawingBox.YMax = Y_Max + Y_Change

End Function

' ************************************************************
```

Create a "ZoomTo" Rect Object Sized a Percentage Larger than the Original Drawing Component Rect

```
Private Function Resize_DrawingBox( _
           ByRef Comp, _
           ByRef X_Percent, _
           ByRef Y_Percent)

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca
```

```
  ' Receives a Drawing Component and enlarges or reduces the
  ' Geoms bounding box by a Percent.

  ' The function creates a Rect Object that is sized X% wider
  ' and Y% higher than the Line drawing object's original
  ' extends.  You can use the new Rect in the "ZoomTo Rect"
  ' MapWindow Method to add some white space around the
  ' primary ZoomTo component.  Just personal opinion but
  ' I think the maps look better zoomed out from the
  ' track lines a bit.
  '
  ' NOTE: In many cases, the Y extend does not seem to
  '       affect how Manifold actually zooms a MapWindow.
  '       X dominates while the changed Y sometimes seems
  '       to be ignored in the final map widow display.
  '       This seems related to where the geographic
  '       objects actually reside on the final map
  '       but I don't have a good handle on this yet.

  '       One could use a single % change for both
  '       width and height and often end result would
  '       be similar.

  ' Example Useage:
  ' ~~~~~~~~~~~~~~~
  ' We want to ZoomTo Component "Lines" but allow a
  ' buffer around the original Lines rectangle.
  '   Set Win_Boundaries = Resize_DrawingBox(Lines, 0.10, 0.05)
  ' Win_Boundaries is now a Rect Object sized 10%
  ' larger in the X direction and 5% larger in the Y
  ' direction than the original Lines bounding box.
  '   Map_Window.ZoomTo(Win_Boundaries)

Dim X_Max, X_Min, Y_Max, Y_Min
Dim X_Change, Y_Change

If Comp.Type <> ComponentDrawing Then
   Application.MessageBoxEX  _
      "Function: Resize_CompBox" & VbCrLf & VbCrLf & _
      "You are attempting to resize the bounding" & VbCrLf & _
      "rectangle for a component that is NOT a" & VbCrLf & _
      "Drawing. This function only works on Drawings.", _
      "Not a Drawing", _
     MessageBoxTypeOK OR MessageBoxTypeIconWarning
   Exit Function
End If

Set Resize_DrawingBox = Comp.ObjectSet.GeomSet.Box

  ' The RectOobject's min and max properties return
  ' values that are in the drawing's projection.

X_Max = Resize_DrawingBox.XMax
X_Min = Resize_DrawingBox.XMin
Y_Max = Resize_DrawingBox.YMax
Y_Min = Resize_DrawingBox.YMin
```

```
   X_Change = (X_Max - X_Min) * X_Percent / 2
   Y_Change = (Y_Max - Y_Min) * Y_Percent / 2

   Resize_DrawingBox.XMin = X_Min - X_Change
   Resize_DrawingBox.XMax = X_Max + X_Change

   Resize_DrawingBox.YMin = Y_Min - Y_Change
   Resize_DrawingBox.YMax = Y_Max + Y_Change

End Function

' ************************************************************************
```

Does a Component Name Exist in The Project?

```
Private Function Component_Exists(ByVal ComponentName)
      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca
   Dim Component_Index
   Dim Comps

   Set Comps = Document.ComponentSet

   Component_Index = Comps.ItemByName(ComponentName)
   If Component_Index < 0 Then
      Component_Exists = False
   Else
      Component_Exists = True
   End If

End Function

' ************************************************************************
```

Delete a Project Component (pass component name)

```
Private Sub Delete_Component(ComponentName)
      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

   Dim Component_Index
   Dim Comps

   Set Comps = Document.ComponentSet

   Component_Index = Comps.ItemByName(ComponentName)
   If Component_Index < 0 Then Exit Sub

   Comps.Remove(Component_Index)

End Sub
```

```
' ************************************************************************
```

"Paste As" Component Type (returns new component Index)

L. Ketch – Originally Posted at:
http://forum.manifold.net/Site/Thread.aspx?id=22518

```
Private Function Document_PasteAs(ByRef ComponentType_Constant)
      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

      ' "ComponentType_Constant" can be one of the following Manifold
      '  ComponentType Constants
      '
      ' ComponentChart    ComponentComments  ComponentDrawing  ComponentElevation
      ' ComponentFolder   ComponentForm      ComponentImage    ComponentLabels
      ' ComponentLayout   ComponentMap       ComponentNull     Invalid value -
Typically used in initialization.
      ' ComponentPalette  ComponentProfile   ComponentQuery    ComponentScript
      ' ComponentSurface  ComponentTable     ComponentTerrain  ComponentZones

      ' Background:
      ' ~~~~~~~~~~
      ' It is common for Manifold programmers to obtain the Index of a new
      ' component pasted from the clipboard by using the Count property of
      ' the ComponentSet in the manner shown below.  In this example, both
      ' the Drawing and its associated table will be pasted (in that order)
      ' and we must subtract 2 from the ComponentSet.Count property.
      '
      '    Drwg.Copy(TRUE)
      '    Document.PasteAs(ComponentDrawing)
      '    Set New_Drawing = Comps.Item(Comps.Count - 2)
      '
      ' There is no guarantee that "(ComponentSet.Count - 2)" will reference the
      ' correct component Index although it generally does. In a GeoReference
      ' thread from June 9, 2006 (adamw at 6/9/2006 7:53 AM (#22465)), Adam
      ' provided a general outline for solving this problem:
      '   "One of the possible ways to locate the components created by the paste
      '    operation (which can create several components!) is to save the IDs of
      '    all existing components prior to doing the paste and locate the new
      '    components after the paste."
      ' see: http://69.17.46.171/Site/Thread.aspx?id=22339&ti=632854363953230000
      '
      ' This Function attempts to implement that solution.
      '
      ' USEAGE:
      ' ~~~~~~
      ' Do NOT use coding such as:
      '    Document.PasteAs ComponentDrawing
      '    Set NewDrawing = Comps.Item(Comps.Count - 2)
      '
      ' USE this instead.
      ' New_Index = Document_PasteAs ComponentDrawing
      ' If New_Index = FALSE Then
```

```
   '      Application.Messagebox "Paste failed"
   '      Exit Sub
   ' End If
   ' Set NewDrawing = Comps.Item(New_Index)

Dim Comp, Comps
Dim Index, New_ID
Dim IDs_BeforePaste()
Dim Count_BeforePaste, Count_AfterPaste

Document_PasteAs = FALSE

Set Comps = Document.ComponentSet

Count_BeforePaste = Comps.Count
ReDim IDs_BeforePaste(Count_BeforePaste - 1)

Index = 0
For Each Comp in Comps
   IDs_BeforePaste(Index) = Comp.ID
   Index = Index + 1
Next

   ' Do some error checking just in case the Paste fails.
Err.Clear
On Error Resume Next
   Document.PasteAs ComponentType_Constant
   If Err Then
      Err.Clear
      Exit Function
   End If
On Error GoTo 0

Count_AfterPaste = Comps.Count

If Count_BeforePaste = Count_AfterPaste Then Exit Function

For Each Comp in Comps
    New_ID = TRUE
    For Index = 0 to (Count_BeforePaste - 1)
        If Comp.ID = IDs_BeforePaste(Index) Then
           New_ID = FALSE
           Exit For
        End If
    Next

    If New_ID Then
             ' We have a newly added ID
         Index = Comps.ItemByID(Comp.ID)
         If Index > -1 Then
            Set Comp = Comps.Item(Index)
                ' Some Paste operations result in more than one new Component
                ' being added to the project. Ex. Pasting a drawing also
                ' pastes the associated table and we get 2 new components.
                ' The user was pasting a primary Component Type. Pass that
                ' Index back.
            If Comp.Type = ComponentType_Constant Then
```

```
                Document_PasteAs = Index
                Exit Function
              End If
            End If
        End If
    Next

End Function

' ************************************************************
```

Duplicate a Component

L. Ketch: Not Posted

```
Option Explicit

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca
      '
      ' Script workaround to find a replacement for the GUI
      ' Duplicate function.  This routine uses a copy and
      ' paste approach.

Sub Main
  Dim Comp, Comps, New_Index

  Set Comps = Document.ComponentSet
  Set Comp = Comps.Item("DrawingB")

  New_Index = Duplicate_Component(Comp, TRUE)
  If New_Index = False Then
    Exit Sub
  Else
    Set Comp = Comps.Item(New_Index)
  End If

  Application.MessageBox "Pasted Component's Name: """ & Comp.Name & """"
End Sub

' ************************************************************************

Private Function Duplicate_Component( _
   ByRef Original_Comp, _
   ByRef Duplicate_Selection)

      ' Comp:      Pass a Manifold Component Object to this routine.
      '            The Comp will be duplicated using a Copy/Paste operation.
      '
      ' Duplicate_Selection: TRUE:  Copy the Component Selection only and
      '                             create a new component based upon that
      '                             selection.  If there is no Selection,
      '                             then the process fails.
      '                     FALSE: Reproduce the entire component.
      '
```

```
' NOTES:
'  1. Not all Components allow a Selection copy. In fact, some
'     do not provide a Copy Method. Summary below.
'  2. Manifold will assign a default name to the copy.
'     The user can change this later if required.
'
' Type       Enumeration             Allows Selection Copy
' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' Chart      ComponentChart          False
' Comments   ComponentComments       False
' Drawing    ComponentDrawing        True
' Elevation  ComponentElevation      False
' Folder     ComponentFolder         No Copy Method
' Form       ComponentForm           No Copy Method
' Image      ComponentImage          True
' Labels     ComponentLabels         True
' Layout     ComponentLayout         True
' Map        ComponentMap            False
' Invalid    ComponentNull           Typically used in initialization.
' Palette    ComponentPalette        True
' Profile    ComponentProfile        False
' Query      ComponentQuery          False
' Script     ComponentScript         False
' Surface    ComponentSurface        True
' Table      ComponentTable          True
' Terrain    ComponentTerrain        False
' Zones      ComponentZones          No Copy Method

' Background:
' ~~~~~~~~~~
' It is common for Manifold programmers to obtain the Index of a new
' component pasted from the clipboard by using the Count property of
' the ComponentSet in the manner shown below.  In this example, both
' the Drawing and its associated table will be pasted (in that order)
' and we must subtract 2 from the ComponentSet.Count property.
'
'    Drwg.Copy(TRUE)
'    Document.PasteAs(ComponentDrawing)
'    Set New_Drawing = Comps.Item(Comps.Count - 2)
'
' There is no guarantee that "(ComponentSet.Count - 2)" will reference the
' correct component Index although it generally does. In a GeoReference
' thread from June 9, 2006 (adamw at 6/9/2006 7:53 AM (#22465)), Adam
' provided a general outline for solving this problem:
'   "One of the possible ways to locate the components created by the paste
'    operation (which can create several components!) is to save the IDs of
'    all existing components prior to doing the paste and locate the new
'    components after the paste."
' see: http://69.17.46.171/Site/Thread.aspx?id=22339&ti=632854363953230000
'
' This Function attempts to implement that solution.
'
' USEAGE:
' ~~~~~~
' Do NOT use coding such as:
'    Component.Copy
'    Document.PasteAs ComponentDrawing
```

```
      '    Set NewDrawing = Comps.Item(Comps.Count - 2)
      '
      ' USE this instead.
      ' New_Index = Duplicate_Component Comp, FALSE
      ' If New_Index = FALSE Then
      '    Exit Sub
      ' Else
      '    Set NewComp = Comps.Item(New_Index)
      ' End If

   Dim Comps, Comp, Qry, Rcrds
   Dim Index, New_ID
   Dim IDs_BeforePaste()
   Dim Count_BeforePaste, Count_AfterPaste
   Dim Selections_Not_Allowed
   Dim Msg

   Msg = "Attempt to duplicate component """ & _
      Original_Comp.Name & """ failed." & VbCrLf & VbCrLf & _
     "Function: Duplicate_Component"

   Duplicate_Component = FALSE

   If Original_Comp.Type = ComponentFolder Or _
      Original_Comp.Type = ComponentForm Or _
      Original_Comp.Type = ComponentNull Or _
      Original_Comp.Type = ComponentZones Then
        Application.MessageboxEX _
           "You are attempting to duplicate """ & _
             Original_Comp.Name & """ (Type: " & Original_Comp.TypeName & ")" & VbCrLf
& _
           "using a Copy and Paste methodology.  However, there" & VbCrLf & _
           "there is no COPY method for this component type." & VbCrLf & VbCrLf & _
           "Function: Duplicate_Component", _
           "Error", MessageBoxTypeOK OR MessageBoxTypeSystemModal
        Exit Function
   End If

   If Original_Comp.Type = ComponentChart OR _
      Original_Comp.Type = ComponentComments OR _
      Original_Comp.Type = ComponentElevation OR _
      Original_Comp.Type = ComponentMap OR _
      Original_Comp.Type = ComponentProfile OR _
      Original_Comp.Type = ComponentQuery OR _
      Original_Comp.Type = ComponentScript OR _
      Original_Comp.Type = ComponentTerrain Then
         Selections_Not_Allowed = TRUE
   Else
         Selections_Not_Allowed = FALSE
   End If

   Set Comps = Document.ComponentSet

     ' create temporary query component
   If Component_Exists("Query") = FALSE Then
      Set Qry = Document.NewQuery("Query")
   Else
```

```
      Set Qry = Comps.Item("Query")
   End If

   Count_BeforePaste = Comps.Count
   ReDim IDs_BeforePaste(Count_BeforePaste - 1)

   Index = 0
   For Each Comp in Comps
      IDs_BeforePaste(Index) = Comp.ID
      Index = Index + 1
   Next

   If Selections_Not_Allowed Then
      Original_Comp.Copy
   ElseIf Duplicate_Selection Then
         ' Is there a selection?  If not, we are done.
      Qry.Text = "Select * From [" & Original_Comp.Name & "] Where [Selection (I)];"
      Qry.RunEx TRUE

      Set Rcrds = Qry.Table.RecordSet
      If Rcrds.Count = 0 Then
         Application.MessageboxEX _
           "You are attempting to create a new " & uCase(Original_Comp.TypeName) &
VbCrLf & _
           "by Pasting SELECTED objects from """ & Original_Comp.Name & """." & VbCrLf
& _
           "However, no objects are currently selected." & VbCrLf & VbCrLf & _
           "Function: Duplicate_Component", _
           "No Selected Records", _
           MessageBoxTypeOK OR MessageBoxTypeSystemModal
         Exit Function
      End If

      Original_Comp.Copy TRUE   ' Copy the Selection
   Else
      Original_Comp.Copy FALSE  ' Copy the entire Drawing
   End If

      ' Do some error checking just in case the Paste fails.
   Err.Clear
   On Error Resume Next
      Document.PasteAs Original_Comp.Type
      If Err Then
         Err.Clear
         Application.MessageboxEX  Msg, "Error",  MessageBoxTypeOK OR
MessageBoxTypeSystemModal
         Exit Function
      End If
   On Error GoTo 0

   Count_AfterPaste = Comps.Count

   If Count_BeforePaste = Count_AfterPaste Then
      Application.MessageboxEX  Msg, "Error",  MessageBoxTypeOK OR
MessageBoxTypeSystemModal
      Exit Function
   End If
```

```
   For Each Comp in Comps
       New_ID = TRUE
       For Index = 0 to (Count_BeforePaste - 1)
           If Comp.ID = IDs_BeforePaste(Index) Then
              New_ID = FALSE
              Exit For
           End If
       Next

       If New_ID Then
                ' We have a newly added ID
            Index = Comps.ItemByID(Comp.ID)
            If Index > -1 Then
               Set Comp = Comps.Item(Index)
                   ' Some Paste operations result in more than one new Component
                   ' being added to the project. Ex. Pasting a drawing also
                   ' pastes the associated table and we get 2 new components.
                   ' The user asked for a duplicate of a specific Component.
                   ' That Index will be passed back.
                   ' Example: User requested a duplicate of a Drawing.
                   ' Pasting the Drawing will also Paste the associated
                   ' Table. We are interested in the Index for the Drawing.
               If Comp.Type = Original_Comp.Type Then
                  Duplicate_Component = Index
                  Exit Function
               End If
            End If
       End If
   Next

   If Duplicate_Component = FALSE Then
            ' If we got to this code are then Duplicate_Component will be FALSE
      Application.MessageboxEX  Msg, "Error",  MessageBoxTypeOK OR
MessageBoxTypeSystemModal
   End If

End Function

' ****************************************************************************

Private Function Component_Exists(ByVal ComponentName)
      ' Author: Lorne Ketch
      ' Permission to use, modify and distribute this code is granted,
      ' provided that you cite the original author.
   Dim Component_Index
   Dim Comps

   Set Comps = Document.ComponentSet

   Component_Index = Comps.ItemByName(ComponentName)
   If Component_Index < 0 Then
      Component_Exists = False
   Else
      Component_Exists = True
   End If
```

```
End Function

' ************************************************************************
```

Close a Window

```
Private Sub Close_Window (ByVal Comp_Name)

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

   Dim Index, LastIndex, WinSet

   Set WinSet = Application.WindowSet

   LastIndex = WinSet.Count - 1

   For Index = 0 to LastIndex
      If uCase(Comp_Name) = uCase(WinSet(Index).Component.Name) Then
          WinSet(Index).Close
          Exit Sub
      End If
   Next

End Sub

' ************************************************************************
```

Retrieve all Component Names

```
Sub Main
  Dim Comps, Comp
  Dim Comment_ComponentNames
  Dim ComponentNames()
  Dim Counter

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

  Set Comps = Document.ComponentSet

  If Component_Exists ("ComponentNames") Then
     Set Comment_ComponentNames = Comps.Item("ComponentNames")
     Comment_ComponentNames.Clear()
  Else
     Set Comment_ComponentNames = Document.NewComments("ComponentNames")
  End If

'  Application.MessageBox Comps.count

  Comment_ComponentNames.AddText _
```

```
        "Document.ComponentSet.Count indicates that" & VbCrLf & _
        "there are " & Comps.count & " Components in this project." & VbCrLf & VbCrLf
  ReDim ComponentNames(300)
  Counter = -1

  For each Comp in Comps
    Counter = Counter + 1
'    Comment_ComponentNames.AddText(Comp.Name)
'    Comment_ComponentNames.AddText VbCrLf
    ComponentNames(Counter) = Comp.Name
    If Counter > Ubound(ComponentNames) Then
       ReDim Preserve ComponentNames (Ubound(ComponentNames) + 100)
    End If
  Next

  Redim Preserve ComponentNames(Counter)
  BubbleSort_uCaseText ComponentNames, FALSE

  For Counter = 0 to Ubound(ComponentNames)
     Comment_ComponentNames.AddText(ComponentNames(Counter))
     Comment_ComponentNames.AddText  VbCrLf
  Next

  Comment_ComponentNames.Open

End Sub

' ************************************************************************
```

Retrieve Open Window Indexes and Names (into dynamic array)

C:\Documents and Settings\Lorne_Limited\My Documents\Manifold\VBscript\Retrieve_OpenWindow_Names.map

```
' ************************************************************************

Option Explicit

' ************************************************************************

Sub Main
   Dim WinNames()
   Dim Index, LastIndex
   Dim Report

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

'   Close_Window "Query"
'   Exit Sub

   Report = ""

   If Window_Names(WinNames) Then
     LastIndex = uBound ( WinNames, 2 )
```

```
     If LastIndex = 0 Then
         Report = "One Window is Currently Open:" & VbCrLf
     Else
         Report = LastIndex + 1 & " Windows are Currently Open:" & VbCrLf & VbCrLf & _
             "Window Index    Component Name" & VbCrLf & _
             "~~~~~~~~~~~~~~~~~~~~~~~~"

     End If

     For Index = 0 to LastIndex
        Report = Report &  VbCrLf &  WinNames(0, Index) & "                          " &
WinNames(1, Index)
     Next

     Application.MessageBox Report

   Else
     Application.MessageBox "No Open Windows"
   End If

End Sub

' ************************************************************************

Private Function Window_Names (ByRef WinNames())
      ' WinNames is dynamic array which will be initialized
      ' in two dimensions. Look at all currently open windows.
      '   Save the WindowSet Number to the 1st dimension.
      '   Save the associated component name to the 2nd dimension.

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

   Dim Index, LastIndex, WinSet

   Window_Names = TRUE
   Set WinSet = Application.WindowSet

   LastIndex = WinSet.Count - 1

   If LastIndex < 0 Then
       Window_Names = False
       Exit Function
   End If

   ReDim WinNames(1, LastIndex)

   For Index = 0 to LastIndex
      WinNames(0, Index) = Index                          ' Window Index
      WinNames(1, Index) = WinSet(Index).Component.Name  ' Open Window for this
Component name
   Next

End Function

' ************************************************************************
```

Close a Window

```
Private Sub Close_Window (ByVal Comp_Name)

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

   Dim Index, LastIndex, WinSet

   Set WinSet = Application.WindowSet

   LastIndex = WinSet.Count - 1

   For Index = 0 to LastIndex
      If Ucase(Comp_Name) = Ucase(WinSet(Index).Component.Name) Then
          WinSet(Index).Close
          Exit Sub
      End If
   Next

End Sub

' ************************************************************************
```

Unselect All Project Components (typically used in Maps)

```
Private Sub UnSelect_All
  Dim Comp, Comps
  Dim Comp_Type

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

     ' General purpose Sub to cancel all selections in
     ' various component types that would often show up.
     ' in Map Components.  This could be made specific
     ' to a given map window with a little extra code.

  Set Comps = Document.ComponentSet

  For Each Comp in Comps
     Comp_Type = Comp.Type

     If Comp_Type = ComponentDrawing OR _
        Comp_Type = ComponentTable OR _
        Comp_Type = ComponentImage OR _
        Comp_Type = ComponentLabels OR _
        Comp_Type = ComponentSurface OR _
        Comp_Type = ComponentTerrain Then
            Comp.SelectNone
     End If
  Next
```

```
End Sub

' ************************************************************************
```

Import Components from Other Projects

```
Option Explict

Sub Main
  Dim Comps, Comp
  Dim Project_Path, External_Doc

       ' Author: Lorne Ketch
       ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

  Project_Path = "C:\Documents and Settings\Lorne_Limited\My
Documents\Manifold\TrackLines\Tracklines.map"
  Set External_Doc = Application.NewDocument(Project_Path, TRUE)
     ' TRUE above = Read only
  Set Comps = External_Doc.ComponentSet
  Set Comp = Comps("Drawing")

  Comp.Copy False ' Copy Everything

  Document.Paste  ' Paste to the current Document.

End Sub

' ************************************************************************
```

Merge GeoBase DEM Tiles

```
' NOTE: Uses Michael Sumner's code in the primary conversion function.
' http://forum.manifold.net/Site/Thread.aspx?id=498&ti=632544163200000000
'
' This script processes DEM tiles downloaded from "GeoBase"
' http://www.geobase.ca/geobase/en/index.html
' Site Map:
'
http://www.geobase.ca/geobase/en/sitemap.html;jsessionid=C7BFB068C87A9127E967BB6EDD989
9DC

' The data is extracted from the National Topgraphic Database (NTDB) files set
' 1:50k tiles are being processed here.
'
' Manifold imports the DEMs fairly accurately.  There are two minor errors that
' have to be corrected by modifying settings in "Edit | Assign Projection"
'
' 1. Manifold sees the DATUM as "North American 1927 (Mean for Conus)" when it
'    should be "North American 1983 (Canada)" - I think :-)
```

```
' 2. The localOffsetY setting is imported 0.75 arc-seconds too low.
'
' NOTE: You must first select the start tile.  Manifold uses a localOffsetX
' and localOffsetY to set the lower left corner of the window.  It might be
' best to choose that tile as the start tile.

' Lorne Ketch
' LKetch@hfx.eastlink.ca
' March 14, 2006
' Update Oct 29, 2006 - Some V7 methods and properties introduced.


Option Explicit

' ****************************************************************************

Sub Main
  Dim Comps, Comp
  Dim Surfaces()
  Dim NumberOfSurfaces
  Dim Primary_Surface

  Set Comps = Document.ComponentSet

       '  You must set the tile component which will be used as the
       '  beginning tile (others will be inserted into this tile)
  Primary_Surface = "021h01_0200_demw"

  NumberOfSurfaces = 0
  Redim Surfaces(100)

  For Each Comp in Comps
     If Comp.Type = ComponentSurface Then
        NumberOfSurfaces =  NumberOfSurfaces + 1
        If NumberOfSurfaces - 1 > uBound(Surfaces) Then
           Redim Preserve Surfaces( uBound(Surfaces) + 100)
        End If
        Surfaces(NumberOfSurfaces - 1) = Comp.Name
     End If
  Next

  Redim Preserve Surfaces(NumberOfSurfaces - 1)

  If NumberOfSurfaces = 0 Then
     Application.MessageBox "There are no Surfaces in this project."
     Exit Sub
  End If

  If NumberOfSurfaces = 1 Then
     Application.MessageBox "There is only One surface in this project."
     Exit Sub
  End If

  Set_CoordSys_Parametres Surfaces

  InsertTiles Surfaces, Primary_Surface
```

```
End Sub

' ************************************************************************

Sub Set_CoordSys_Parametres (ByRef Surfaces())
   ' GeoReference code provided by Michael D. Sumner
   ' Date: December 10 2003 Version: 1.0.1
   ' http://forum.manifold.net/Site/Thread.aspx?id=498&ti=632544163200000000


  Dim Counter
  Dim Comp, Comps
  Dim CoordSys, CoordSys_Parametres
  Dim CorrectProjection, CorrectDatum
  Dim local_ScaleX, local_ScaleY, local_OffsetX, local_OffsetY
  Dim NumberOfSurfaces

  CorrectProjection = "Latitude / Longitude"
  CorrectDatum = "North American 1983 (Canada)"

  Set Comps = Document.ComponentSet

  NumberOfSurfaces = uBound(Surfaces) + 1

  For Counter = 0 to NumberOfSurfaces - 1
     Set Comp = Comps(Surfaces(Counter))
     Set CoordSys = Comp.CoordinateSystem
     Set CoordSys_Parametres = CoordSys.Parameters
     Application.StatusText = "Setting Current Projection: " &  Counter + 1 & " of " &
NumberOfSurfaces

         ' The NTDB tiles import fine except for the DATUM and y offset
         ' Save the original values for offset and scale so they
         ' can be reset further down.

         ' Thes parameters below may also be referenced by
         ' name using
CoordSys_Parametres(CoordSys_Parametres.ItemByName("localScaleX"))

     local_ScaleX  = CoordSys_Parametres.Item("localScaleX").Value
     local_ScaleY  = CoordSys_Parametres.Item("localScaleY").Value
     local_OffsetX = CoordSys_Parametres.Item("localOffsetX").Value
     local_OffsetY = CoordSys_Parametres.Item("localOffsetY").Value

'     CoordSys.Load(CorrectProjection)
     CoordSys.Datum.Load(CorrectDatum)

     CoordSys_Parametres.Item("localScaleX").Value  = local_ScaleX
     CoordSys_Parametres.Item("localScaleY").Value  = local_ScaleY
     CoordSys_Parametres.Item("localOffsetX").Value = local_OffsetX
     CoordSys_Parametres.Item("localOffsetY").Value = local_OffsetY + 0.75

     Comp.CoordinateSystemVerified = TRUE

  Next

End Sub
```

```
' ********************************************************************

Sub InsertTiles (ByRef Surfaces(), ByRef Main_Surface)

'   L. Ketch

    ' Surfaces():   Dynamic array holding the Surface Component Names
    ' MainSurface:  The Start Tile name.  Other tiles will be pasted
    '               into this one.
  Dim Comp, Comps
  Dim TargetTile
  Dim Counter
  Dim TileCounter
  Dim NumberOfSurfaces

  Set Comps = Document.ComponentSet
  Set TargetTile = Comps(Main_Surface)

  NumberOfSurfaces = uBound(Surfaces) + 1

  TileCounter = 1

  For Counter = 0 to uBound(Surfaces)

     If uCase(Main_Surface) <> uCase(Surfaces(Counter)) Then
        Set Comp = Comps(Surfaces(Counter))
        Application.StatusText = "Inserting Tile: " &  TileCounter & " of " &
NumberOfSurfaces
        TileCounter =  TileCounter + 1

        Comp.Copy FALSE          'FALSE = Copy the entire component (ignore any
selections)
        TargetTile.Paste FALSE   'FALSE = Paste into the main tile but do not replace
the selection

          ' Delete the component just merged with the main tile
        Comps.Remove Comps(Surfaces(Counter))
     End If

  Next

End Sub

' ********************************************************************
```

Using Regular Expressions

Example taken from: Plot_Cyclone_FcstTrack.map
L. Ketch
C:\Documents and Settings\Lorne_Limited\My Documents\Manifold\Tropical_Cyclone

```
Option Explicit

' ************************************************************************

Sub Main
      ' Assumption
      ' ~~~~~~~~~~
      ' All secondary component names will begin with an 8 character
      ' identifier beginning with either WO or WT and followed by a
      ' 6 digits DDHHMM stamp followed by an underscore...
      ' ex. WT231437_ or WO030451_

   Dim Comp, Comps
   Dim Counter
   Dim Components2Delete()
   Dim Index, LastIndex
   Dim regEx_WO, regEX_WT
   Dim Comp_Name
   Dim Answer

   Answer = _
     Application.MessageBoxEx( _
         "NOTE!" & VbCrLf & VbCrLf & _
         "YOU ARE ABOUT TO DELETE ALL COMPONENTS" & VbCrLf & _
         "that begin with either ""WT"" or ""WO"" followd by" & VbCrLf & _
         "a 6-digit date/time stamp, followed by an underscore."  & VbCrLf & VbCrLf &
_
         "Example: WO272156_ALBERTO_LINES"  & VbCrLf & VbCrLf & _
         "DO YOU WANT TO DO THIS (YES)?",  _
         "Warning!", _
         MessageBoxTypeYesNo OR MessageBoxTypeSystemModal OR
MessageBoxTypeIconWarning)

   If Answer = MessageBoxResultNO Then Exit Sub

   Set regEX_WT = New RegExp
     regEX_WT.IgnoreCase = TRUE
     regEX_WT.Global = TRUE
     regEX_WT.Pattern = "^WT\d\d\d\d\d\d_"

   Set regEX_WO = New RegExp
     regEX_WO.IgnoreCase = TRUE
     regEX_WO.Global = TRUE
     regEX_WO.Pattern = "^WO\d\d\d\d\d\d_"

   Set Comps = Document.ComponentSet

     ' We are constantly deleting components so we have to be
     ' careful. Deleting one component may in fact delete
     ' a number of linked components.  First collect the names
     ' of the compenents to delete, then do the actual deletions
     ' by component name. If the name no longer exists, we can
     ' just ignore it.

   ReDim Components2Delete(100)
   Index = -1
```

```
   For Each Comp in Comps
      Comp_Name = uCase(Comp.Name)

      If regEX_WO.test(Comp_Name) = TRUE Or _
         regEX_WT.test(Comp_Name) = TRUE Then
           Index = Index + 1
           If Index > uBound(Components2Delete) Then
              ReDim Preserve Components2Delete(uBound(Components2Delete)+100)
           End If
           Components2Delete(Index) = Comp.Name
      End If
   Next

   LastIndex = Index
   For Index = 0 to LastIndex
       Delete_Component(Components2Delete(Index))
   Next

End Sub

' ************************************************************************

Private Sub Delete_Component(ComponentName)
   Dim Component_Index
   Dim Comps

   Set Comps = Document.ComponentSet

   Component_Index = Comps.ItemByName(ComponentName)
   If Component_Index < 0 Then Exit Sub

   Comps.Remove(Component_Index)

End Sub

' ************************************************************************
```

Get the Last Token in a String (in Proper case)

```
Private Function Get_Last_Token (ByVal Str, ByVal ParseChar)
  Dim Ptr

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

  Ptr =  InStrRev(Str, ParseChar)
  If Ptr < 1 Then Ptr = 0

  Get_Last_Token = Trim(  Right(Str, Len(Str) - Ptr )  )

     ' Change to Proper case.  No function for this in vbScript
  Get_Last_Token = lCase(Get_Last_Token)
  Get_Last_Token = uCase(Left(Get_Last_Token, 1)) & Right(Get_Last_Token,
Len(Get_Last_Token) - 1)
```

```
End Function

' ************************************************************************
```

BubbleSort Text (based on Upper Case text evaluation)

```
Private Sub BubbleSort_uCaseText (ByRef Arr, ByRef Descending)
     '  BubbleSort - Sort Arrays Using The Bubblesort Algorithm
     '  Date: 9/4/1999
     '  Author: The VB2TheMax Team
     '
     '  BubbleSort is especially convenient with small arrays (1,000
     '  items or fewer) or with arrays that are already almost sorted.
     '
     '  Converted to VBscript by L. Ketch... some modifications as well.

   Dim Value, Index, First_Item, Index_Limit, LastSwap

     ' account for optional arguments
   First_Item = LBound(arr)
   LastSwap   = UBound(arr)

   Do
       Index_Limit = LastSwap - 1
       LastSwap = 0
       For Index = First_Item To Index_Limit

           Value = Arr(Index)

           If (uCase(Value) > uCase(Arr(Index + 1))) Xor Descending Then
                 ' If the items are not in order, swap them
               Arr(Index) = Arr(Index + 1)
               Arr(Index + 1) = Value
               LastSwap = Index
           End If

       Next
   Loop While LastSwap

End Sub

' ************************************************************************
```

Find the Number of Days in any Month (including Feb Leap Years)

```
Private Function MaxDaysInMonths( ByRef Mon, Yr)
     ' NOTE: Calls the IsLeapYear function which must
     '       also be available.
     '
     ' Mon = 1 to 12
```

```
    ' Yr: required if Mon = February.  Checks for Leap year
    '     If month <> February, then just pass any number for the year
    ' L. Ketch

  Dim DaysInMonth(11)

  DaysInMonth(0)  = 31  :  DaysInMonth(1)  = 28  :  DaysInMonth(2)  = 31
  DaysInMonth(3)  = 30  :  DaysInMonth(4)  = 31  :  DaysInMonth(5)  = 30
  DaysInMonth(6)  = 31  :  DaysInMonth(7)  = 31  :  DaysInMonth(8)  = 30
  DaysInMonth(9)  = 31  :  DaysInMonth(10) = 30  :  DaysInMonth(11) = 31

  If Mon = 2 and IsLeapYear(Yr) Then
       ' See IsLeapYear below
     DaysInMonth(1)  = 29
  End If

  MaxDaysInMonths = DaysInMonth(Mon-1)

End Function

' *************************************************************
```

Is Year a Leap-Year

```
Private Function IsLeapYear(ByVal Yr)
    ' Multiples of 100 that are not also multiples of 400 are not
    ' leap years; thus, 1900 was not, but 2000 was.

    ' Original Source:
    ' Woody's Office Watch - March 25, 1998 Vol 3 No 12
    ' http://office-watch.com/office/archtemplate.asp?v3-n12

    If Yr < 100 Or Yr > 9999 Then
        Application.Messagebox "The year provided must be between 100 and 9999,
inclusive."
        Exit Function
    End If

    If Yr Mod 400 = 0 Then
        IsLeapYear = True
    ElseIf Yr Mod 100 = 0 Then
        IsLeapYear = False
    ElseIf Yr Mod 4 = 0 Then
        IsLeapYear = True
    Else
        IsLeapYear = False
    End If

End Function

' *************************************************************************
```

Using SendKeys to Load Dialogs

chrismarx on 11/9/2006 9:59 AM (#30049)

http://forum.manifold.net/Site/Thread.aspx?id=29979

Check this out – here's a way to bypass the dialog box and automate the Paste As process - it sends the "~" command, which is apparently interpreted as an "enter", so as long as your columns are formated with the right names, there's no need to manually work through the dialog box.

```
Sub Main
  Set ActiveDoc = Application.ActiveDocument
  Set CompSet = ActiveDoc.ComponentSet
  Set Shell = CreateObject("WScript.Shell")
  Set Table = CompSet.Item("Sheet1$ 4")
  Table.Copy()
  Shell.SendKeys "~"
  ActiveDoc.PasteAs(ComponentDrawing)
End Sub
```

Lorne on 11/30/2006 4:36 PM (#31271)

(in response to a thread initiated by brbkr
http://forum.manifold.net/Site/Thread.aspx?id=30787&ti=633024219241100000

```
Option Explicit

' SelectNone
' http://69.17.46.171/Site/Thread.aspx?id=21452&ti=632838853471300000
'
' Use of Document and ActiveDocument in V7
' http://69.17.46.171/Site/Thread.aspx?id=21589&ti=632840185176300000

Sub Main
   Dim CompSet, Drwg, Shell, Qry
   Dim LastSurface, I, ActiveDoc_Path
   Dim Sfc, surfaceCmpst

   Set CompSet = Document.ComponentSet
   Set Drwg = CompSet.Item("SalesData Drawing")
   Set surfaceCmpst = CompSet.Item("Composite Surface")

   Set Shell = CreateObject("WScript.Shell")

   ' ActiveDoc_Path = Document.Path

      ' Make sure the Query component exists
   If CompSet.ItemByName("TmpQuery") < 0 Then
      Set Qry = Document.NewQuery("TmpQuery")
   Else
      Set Qry = CompSet.Item("TmpQuery")
   End If
```

```
   If CompSet.ItemByName("SalesData Drawing 2") >= 0 Then
      CompSet.Remove(CompSet.ItemByName("SalesData Drawing 2"))
   End If

   LastSurface = 47
   For I = 1 to LastSurface

      Qry.Text = _
        "UPDATE [SalesData Drawing] SET [SalesData Drawing].[Selection (I)]=TRUE " & _
            "WHERE Contains((SELECT [Neighborhood Buffer Drawing].[Geom (I)] " & _
            "FROM [Neighborhood Buffer Drawing] WHERE [NeighbNum] = " & cStr(I) & "),"
& _
        "[SalesData Drawing].[Geom (I)])"

      Qry.Run

      Drwg.Copy TRUE

      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "100"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys "{TAB}"
      Shell.SendKeys " "
      Shell.SendKeys "~"

      Document.PasteAs(ComponentSurface)

      Set Sfc = CompSet.Item("SalesData Drawing 2")

      Qry.Text = _
         "UPDATE (SELECT [SalesData Drawing 2].[Selection (I)] " & _
           "FROM [Neighborhood Drawing],[SalesData Drawing 2] " & _
           "WHERE Contains([Neighborhood Drawing].[ID], " & _
           "NewPointLatLon([SalesData Drawing 2].[Longitude (I)],[SalesData Drawing
2].[Latitude (I)])) " & _
           "AND [Neighborhood Drawing].[NeighbNum] = " & cStr(I) & ") SET [SalesData
Drawing 2].[Selection (I)] = TRUE"
      Qry.Run

      Sfc.Copy TRUE

      Application.StatusText = "Pasting Surface #" & cStr(I) & " of " &
cStr(LastSurface)

      surfaceCmpst.Paste TRUE

      surfaceCmpst.SelectNone
```

```
      Drwg.SelectNone

      CompSet.Remove(CompSet.ItemByName("SalesData Drawing 2"))

         ' SaveAs below will save the project from within this
         ' loop but appears to cause a problem later - after the
         ' the script has finished.
      'If I < 4 Then Document.SaveAs(ActiveDoc_Path)

   Next

   CompSet.Remove(CompSet.ItemByName("TmpQuery"))

End Sub
```

chrismarx on 12/22/2006 10:05 PM (#32674)

I've found that I sometimes get different results if I run the script under the debugger, or without -- sometimes one is better than other. Also, you may want to include some "sleep" statements between some of sendkeys, that may help

Here's a little function you can put into your script.

```
Function Sleep(t)
  dim dtGoal
  dtgoal = DateAdd("s", t, Now)
  do while dtgoal > Now
  Loop
End Function
```

I usually call sleep(.5) if I feel that particular menu or whatever needs sometime. However, I've found that in some cases, putting in this sleep actually wrecks "the flow" of the sendkey command, --- and yet other times its vital. just another thing to play with!

You Cannot Instantiate the Root of the WSH Object Model (WScript)

brbkr on 9/26/2005 10:39 AM (#15052)

What I am trying to do, is use "SendKeys" commands from a script to accomplish something that can only be done through the Manifold GUI (See WillH's 4/14/05 post in "General Scripting Questions" regarding the "PasteAs" command and importing a table as a drawing.) But I can't seem to directly reference the WScript root object of the Windows Script Host Object Model. I can do the following and pull down the file menu as desired:

```
Sub Main
    Set WshShell = CreateObject("WScript.Shell")
    WshShell.SendKeys "%f"
End Sub
```

But if I use the prescribed code, as follows, I get the error "Object required: 'WScript' Line:2"

```
Sub Main
    Set WshShell = WScript.CreateObject("WScript.Shell")
    WshShell.SendKeys "%f"
    WScript.Sleep 500
End Sub
```

Using the former code would be OK, except I need to reference the WScript object to use the WScript.Sleep command and make the code execute properly. Any help would be greatly appreciated!

chrismarx on 12/22/2006 12:42 PM (#32659)

Better late than never. Your answer is here

http://blogs.msdn.com/ericlippert/archive/2003/10/08/53175.aspx#53176

Basically, ***if you're working in manifold, you can't instantiate the root of the WSH object model*** (wscript) or any of its associated methods-

Casting Geom (I) – Binary Column Data Not Equal a GEOM Object

http://forum.manifold.net/Site/Thread.aspx?id=32618&ti=633023275408900000

Robotuner on 12/21/2006 6:02 PM (#32619)

I am trying to get the `Geom (I)` object in c# by:

```
Double x=((Manifold.Interop.Geom)rs.get_Data("Geom (I)").Center.X
```

where `rs` is a `Manifold.Interop.Record`

I keep getting the error message:
Unable to cast object of type `'System.Byte[]'` to type `'Manifold.Interop.Geom'`.

Is there a work around?

Thanks.

willh on 12/21/2006 6:29 PM (#32621)

You're on your own with C#, but I can tell you that the binary data stored in the column doesn't equal a geom object. Use something like this to get a geom object (VB)

```
Application.NewGeomFromBinary(Record.Data("geom (i)"))
```

Also, I would venture a guess that there are much better ways to get a geom than this. Just for one you could use record.object.geom

Robotuner on 12/21/2006 7:52 PM (#32625)

thanks a bundle, your suggestions were enough for me to figure out the correct syntax:

```
double x1=((Manifold.Interop.Object)rs.Object).get_Geom().Center.X;
```

c# can be a real pain sometimes!

NewImport Problems – External Database Connection Fails

http://forum.manifold.net/Site/Thread.aspx?id=30605

(This may be fixed in Build: 7.1.14.917 – L. Ketch)

adamw on 1/15/2007 8:16 AM (#33481)

The ADO .NET imports do not work due to a bug. I have filed a request to fix the bug in the nearest update.

The OLE DB import seems to work:

```
'VB.NET
Imports M = Manifold.Interop
Imports Manifold.Interop.Scripts

Class Script
  Shared Sub Main
    Dim iadonet As M.ImportOleDb = Context.Application.NewImport("OLEDB")
    iadonet.ConnectionString = "Provider=SQLOLEDB.1;" & _
      "Integrated Security=SSPI;" & _
      "Persist Security Info=False;" & _
      "Initial Catalog=test;" & _
      "Data Source=test"
    iadonet.ConvertPolicy = M.ConvertPolicy.ConvertSome
      ' ^^ use the Tables property
    iadonet.Tables.Add("dbo.Mexico")
    iadonet.Import("") ' ignore the parameter
  End Sub
End Class
```

Export a Palette using a Script

http://forum.manifold.net/Site/Thread.aspx?id=33341

adamw on 1/18/2007 6:17 AM (#33617)

You can export a palette using a script:

```
' VBScript
Sub Main
```

```
  Set pal = Document.ComponentSet("Palette")
  Set cmt = Document.NewComments("Palette Dump")
  For Each color In pal.ColorSet
    cmt.AddText(color.Red & ":" & color.Green & ":" & color.Blue & Chr(13) & Chr(10))
  Next
  cmt.Open
End Sub
```

Preset Definintions: Command Filter List

http://69.17.46.171/Site/Thread.aspx?id=20620&ti=632823354777800000

adamw at 5/4/2006 6:33 AM (#20838)

Invoke the command prompt, navigate to the Manifold installation folder and run the following command line: manifold.exe /clist:path.txt, where "path.txt" is somewhere you have write access to. This will dump the names of all available commands.

List for V7x

ChartData
ChartDataMoveDown
ChartDataMoveToBottom
ChartDataMoveToTop
ChartDataMoveUp
ChartDataSelectAll
ChartDataSelectInverse
ChartDataSelectNone
ChartDataShowTypes
ChartFormat
ChartFormatInterpolate
ChartFormatLoad
ChartFormatLoadFromFile
ChartFormatReverse
ChartFormatSaveToFile
ChartLabels
ChartLogarithmic
DrawingColor
DrawingDissolve
DrawingDistricts
DrawingDistrictsAdvanced
DrawingDistrictsVisual
DrawingDistrictsVisualAdd
DrawingDistrictsVisualClear
DrawingDistrictsVisualClearAll
DrawingDistrictsVisualDelete
DrawingDistrictsVisualFill
DrawingDistrictsVisualMoveDown
DrawingDistrictsVisualMoveToBottom
DrawingDistrictsVisualMoveToTop
DrawingDistrictsVisualMoveUp
DrawingDistrictsVisualShowCounts
DrawingDistrictsVisualShowValuePercentages
DrawingDistrictsVisualShowValues
DrawingDriveTimeZones
DrawingMinicharts
DrawingMinichartsMoveDown
DrawingMinichartsMoveToBottom
DrawingMinichartsMoveToTop
DrawingMinichartsMoveUp
DrawingMinichartsSelectAll
DrawingMinichartsSelectInverse
DrawingMinichartsSelectNone
DrawingMinichartsShowTypes
DrawingObjectBranchDelete
DrawingObjectBranchDuplicate
DrawingObjectBranchSnap
DrawingObjectBranchSplit
DrawingObjectCenter
DrawingObjectColumns
DrawingObjectColumnsClear
DrawingObjectColumnsPrint
DrawingObjectColumnsShowBlanks
DrawingObjectColumnsShowIntrinsics
DrawingObjectColumnsShowReadOnly
DrawingObjectColumnsShowTypes
DrawingObjectCoordinateAdd
DrawingObjectCoordinateAddMidSegment
DrawingObjectCoordinateAddOnSegment
DrawingObjectCoordinateDelete
DrawingObjectCoordinateDeleteSplitBranch
DrawingObjectCoordinateDuplicate
DrawingObjectCoordinateSnap
DrawingObjectCoordinates
DrawingObjectCoordinatesAddBranch
DrawingObjectCoordinatesAddPoint
DrawingObjectCoordinatesBreakBranch
DrawingObjectCoordinatesDelete
DrawingObjectCoordinatesDeleteBranch
DrawingObjectCoordinatesMoveDown
DrawingObjectCoordinatesMoveToBottom
DrawingObjectCoordinatesMoveToTop
DrawingObjectCoordinatesMoveUp
DrawingObjectCoordinatesNative
DrawingObjectCoordinatesPrint
DrawingObjectCopy
DrawingObjectCut
DrawingObjectDelete
DrawingObjectDuplicate
DrawingObjectOrthogonalize
DrawingObjectPaste
DrawingObjectPasteAppend
DrawingObjectSegmentDelete
DrawingObjectSegmentDeleteSplitBranch
DrawingObjectSegmentize
DrawingObjectSimplify
DrawingObjectSnap
DrawingObjectZoom
DrawingOpenDataSource
DrawingOpenTable
DrawingOptimalRoute
DrawingOptimalRouteVisual
DrawingOptimalRouteVisualCenter
DrawingOptimalRouteVisualDelete
DrawingOptimalRouteVisualMoveDown
DrawingOptimalRouteVisualMoveToBottom
DrawingOptimalRouteVisualMoveToTop
DrawingOptimalRouteVisualMoveUp
DrawingOptimalRouteVisualPing
DrawingOptimalRouteVisualShowNames
DrawingOptimalRouteVisualTraceRoute
DrawingOrthogonalize
DrawingRelink
DrawingSegmentize
DrawingSendEmail
DrawingSimplify
DrawingSnap
DrawingSpatialOverlay
DrawingTopologyFactory
DrawingTopologyFactoryCenter
DrawingTopologyFactoryFix
DrawingTopologyFactoryFixMethod
DrawingTopologyFactoryFixSelected
DrawingTopologyFactoryGoFirst
DrawingTopologyFactoryGoLast
DrawingTopologyFactoryGoNext
DrawingTopologyFactoryGoPrevious
DrawingTopologyFactorySortAlphabetic
DrawingTopologyFactorySortAlphabeticBlanksLast
DrawingTopologyFactorySortRecommended
DrawingTopologyFactorySortRecommendedBlanks Last
DrawingTopologyOverlay
DrawingTransformAttachTo
DrawingTransformAttachToSelf
DrawingTransformBorderBuffers
DrawingTransformBoundaries
DrawingTransformBoundedAreas
DrawingTransformBoundingBoxes
DrawingTransformBuffers
DrawingTransformCentroids
DrawingTransformCentroidsBox
DrawingTransformCentroidsInner
DrawingTransformCentroidsWeight
DrawingTransformClipWithIntersect
DrawingTransformClipWithSubtract
DrawingTransformClusters
DrawingTransformClustersZahn
DrawingTransformCommonBoundingBox
DrawingTransformCommonBuffer
DrawingTransformCommonCentroid
DrawingTransformCommonEnclosingCircle
DrawingTransformCommonEnclosingRectangle
DrawingTransformConvexHull
DrawingTransformDecompose
DrawingTransformDecomposeToConvexParts
DrawingTransformDecomposeToTriangles
DrawingTransformDistanceNetwork
DrawingTransformEnclosingCircles
DrawingTransformEnclosingRectangles
DrawingTransformExplode
DrawingTransformFarthestNeighbor
DrawingTransformFarthestNeighborSymm

DrawingTransformFarthestPair
DrawingTransformFlipHorizontally
DrawingTransformFlipVertically
DrawingTransformGabrielNetwork
DrawingTransformInnerBuffers
DrawingTransformIntersectLines
DrawingTransformIntersectionPoints
DrawingTransformJoinLines
DrawingTransformMoveHorizontally
DrawingTransformMoveVertically
DrawingTransformNearestNeighbor
DrawingTransformNearestNeighborSymm
DrawingTransformNearestPair
DrawingTransformNodePoints
DrawingTransformNormalizeMetric
DrawingTransformNormalizeTopology
DrawingTransformPoints
DrawingTransformRelativeNeighborhoodNetwork
DrawingTransformRemoveDuplicates
DrawingTransformReverseLines
DrawingTransformRotate
DrawingTransformScale
DrawingTransformScaleHorizontally
DrawingTransformScaleVertically
DrawingTransformSegments
DrawingTransformSelectAdjacentTo
DrawingTransformSelectContainedWithin
DrawingTransformSelectContaining
DrawingTransformSelectCriticalServiceCenter
DrawingTransformSelectEuclideanPointCoverage
DrawingTransformSelectIntersecting
DrawingTransformSelectLineCoverage
DrawingTransformSelectNCriticalServiceCenters
DrawingTransformSelectNServiceCenters
DrawingTransformSelectPointCoverage
DrawingTransformSelectServiceCenter
DrawingTransformSelectShortestPath
DrawingTransformSelectSpanningTree
DrawingTransformSelectTouching
DrawingTransformShapeHull
DrawingTransformSpanningTree
DrawingTransformSpline
DrawingTransformSplit
DrawingTransformTriangulation
DrawingTransformTriangulationAreas
DrawingTransformTriangulationLines
DrawingTransformUnion
DrawingTransformVoronoi
DrawingTransformVoronoiAreas
DrawingTransformVoronoiLines
DrawingTransformVoronoiPoints
DrawingUnlink
EditAlignBottom
EditAlignLeft
EditAlignRight
EditAlignTop
EditAssignProjection
EditBookmarkNext
EditBookmarkPrevious
EditBookmarkToggle
EditBookmarksDeleteAll
EditBreakpointNext
EditBreakpointPrevious
EditBreakpointToggle
EditBreakpointsDeleteAll
EditCenterHorizontally
EditCenterVertically
EditChangeProjection
EditCommentSelection
EditCompleteWord
EditCompleteWordWithList
EditCopy
EditCreateAreas
EditCreateLines
EditCreatePoints
EditCut
EditDelete
EditDeleteAll
EditDeleteInLayers
EditFind
EditFindNext
EditFormatApply
EditFormatAreaBackground
EditFormatAreaBorderBackground
EditFormatAreaBorderForeground
EditFormatAreaBorderSize
EditFormatAreaBorderStyle
EditFormatAreaForeground
EditFormatAreaSize
EditFormatAreaStyle
EditFormatBrushBackground
EditFormatBrushForeground
EditFormatBrushSize
EditFormatBrushStyle
EditFormatDarken
EditFormatElementBackground
EditFormatElementFont
EditFormatElementFontBold
EditFormatElementFontItalic
EditFormatElementFontSize
EditFormatElementFontStrikeout
EditFormatElementFontUnderline
EditFormatElementForeground
EditFormatElementRotation
EditFormatElementTextAlignBottom
EditFormatElementTextAlignLeft
EditFormatElementTextAlignRight
EditFormatElementTextAlignTop
EditFormatElementTextCenterHorizontally
EditFormatElementTextCenterVertically
EditFormatElementTextJustify
EditFormatGrayscale
EditFormatInterpolate
EditFormatLabelAlignBottom
EditFormatLabelAlignLeft
EditFormatLabelAlignRight
EditFormatLabelAlignTop
EditFormatLabelBackground
EditFormatLabelBorder
EditFormatLabelCenterHorizontally
EditFormatLabelCenterVertically
EditFormatLabelFont
EditFormatLabelFontBold
EditFormatLabelFontItalic
EditFormatLabelFontSize
EditFormatLabelFontStrikeout
EditFormatLabelFontUnderline
EditFormatLabelForeground
EditFormatLabelRotation
EditFormatLabelStyle
EditFormatLabelTextAlignLeft
EditFormatLabelTextAlignRight
EditFormatLabelTextCenter
EditFormatLabelTextJustify
EditFormatLighten
EditFormatLineBackground
EditFormatLineForeground
EditFormatLineSize
EditFormatLineStyle
EditFormatLoadFromFile
EditFormatMoveDown
EditFormatMoveToBottom
EditFormatMoveToTop
EditFormatMoveUp
EditFormatPointBackground
EditFormatPointForeground
EditFormatPointRotation
EditFormatPointSize
EditFormatPointStyle
EditFormatReverse
EditFormatSaveToFile
EditFormatTraceColor
EditFullHeight
EditFullSize
EditFullWidth
EditGoBlank
EditGoFirst
EditGoLast
EditGoNext
EditGoPrevious
EditGoTo
EditGoToMatchingBracket
EditHelpOnWord
EditInstantData
EditJoinWrapSelection
EditLoadMaskChannel
EditMakeLowercase
EditMakeUppercase
EditPaintAreas
EditPaintLines
EditPaintPoints
EditPaste
EditPasteAppend
EditPasteAutoselect
EditPasteSelectAll
EditPasteSelectInverse
EditPasteSelectNone
EditProjectionLoadFromComponent
EditProjectionLoadFromFile
EditProjectionLoadRecent
EditProjectionSaveToFile
EditQueryArgument
EditQueryColumn
EditQueryOperation
EditQueryOperationNot
EditQuerySelect
EditRedo
EditRemoveLine
EditReplace
EditResetFormat
EditSameHeight
EditSameHeightMax
EditSameHeightMin
EditSameWidth
EditSameWidthMax
EditSameWidthMin
EditSaveMaskChannel
EditSelectAdd

EditSelectAll
EditSelectAllInLayers
EditSelectAreas
EditSelectBorder
EditSelectByType
EditSelectContract
EditSelectExpand
EditSelectIntersect
EditSelectInverse
EditSelectInverseInLayers
EditSelectInvert
EditSelectLabels
EditSelectLines
EditSelectNone
EditSelectPixels
EditSelectPoints
EditSelectReplace
EditSelectSmooth
EditSelectStyle
EditSelectStyleBorder
EditSelectStyleDashes
EditSelectStyleDenseDots
EditSelectStyleMediumDots
EditSelectStyleNone
EditSelectStyleSparseDots
EditSelectSubtract
EditSelectToMatchingBracket
EditSharedEdit
EditSnapToAreas
EditSnapToColor
EditSnapToElements
EditSnapToGraticule
EditSnapToGrid
EditSnapToLabels
EditSnapToLines
EditSnapToPoints
EditSnapToSegments
EditSnapToSelection
EditSpaceAcross
EditSpaceDown
EditTransformApply
EditTransformOperation
EditTransformSource
EditTransformTarget
EditUncommentSelection
EditUndo
EditWrapSelection
ElevationOpenProfile
ElevationOpenSurface
FileClose
FileCloseAll
FileCreateChart
FileCreateChartMoveDown
FileCreateChartMoveToBottom
FileCreateChartMoveToTop
FileCreateChartMoveUp
FileCreateChartSelectAll
FileCreateChartSelectInverse
FileCreateChartSelectNone
FileCreateChartShowTypes
FileCreateComments
FileCreateDrawing
FileCreateElevation
FileCreateFolder
FileCreateForm
FileCreateImage
FileCreateLabels
FileCreateLayout
FileCreateMap
FileCreateMapMoveDown
FileCreateMapMoveToBottom
FileCreateMapMoveToTop
FileCreateMapMoveUp
FileCreateMapSelectAll
FileCreateMapSelectInverse
FileCreateMapSelectNone
FileCreatePalette
FileCreateProfile
FileCreateQuery
FileCreateScript
FileCreateSurface
FileCreateTable
FileCreateTableAdd
FileCreateTableDelete
FileCreateTableMoveDown
FileCreateTableMoveToBottom
FileCreateTableMoveToTop
FileCreateTableMoveUp
FileCreateTerrain
FileCreateTheme
FileCreateZones
FileExit
FileExportDrawing
FileExportImage
FileExportSelectAll
FileExportSelectInverse
FileExportSelectNone
FileExportSurface
FileExportTable
FileExportText
FileExportWebPage
FileImportAdd
FileImportComments
FileImportComponent
FileImportComponentFromServer
FileImportDelete
FileImportDrawing
FileImportImage
FileImportMoveDown
FileImportMoveToBottom
FileImportMoveToTop
FileImportMoveUp
FileImportQuery
FileImportRefresh
FileImportScript
FileImportSelectAll
FileImportSelectInverse
FileImportSelectNone
FileImportSurface
FileImportTable
FileLinkComments
FileLinkComponent
FileLinkComponentFromServer
FileLinkDrawing
FileLinkImage
FileLinkQuery
FileLinkScript
FileLinkSurface
FileLinkTable
FileNew
FileOpen
FileOpenRecent
FilePageSetup
FilePrint
FilePrintDirect
FileSave
FileSaveAs
FormAbout
FormClose
FormControlBestFit
FormControlCopy
FormControlCut
FormControlDelete
FormControlMoveDown
FormControlMoveToBottom
FormControlMoveToTop
FormControlMoveUp
FormControlPaste
FormControlPasteAppend
FormControlProperties
FormControlSnap
FormOpenScript
FormProperties
FormRun
FormRunUnderDebugger
FormSizeToFit
FormTabOrder
FormTabOrderMoveDown
FormTabOrderMoveToBottom
FormTabOrderMoveToTop
FormTabOrderMoveUp
FormTest
HelpAbout
HelpActivate
HelpActivateExtension
HelpCheckForUpdates
HelpContents
HelpIndex
HelpSearch
HelpWebCommunity
HelpWebFreeStuff
HelpWebHome
HelpWebNews
HelpWebSupport
ImageBrightnessContrast
ImageColorBalance
ImageColorize
ImageConvertTo
ImageCreateIndexDrawing
ImageDiffuse
ImageDither
ImageDownload
ImageEqualize
ImageFilter
ImageFluoresce
ImageGamma
ImageGaussianBlur
ImageHueSaturation
ImageInvert
ImageMotionBlur
ImageNoise
ImageOpenDataSource
ImageOpenPalette
ImagePosterize
ImageQuantize
ImageRelief
ImageRelink
ImageResize

ImageSimplify
ImageThreshold
ImageThresholdColor
ImageTile
ImageTransformAddMargin
ImageTransformAddNoise
ImageTransformAddNoiseMono
ImageTransformAutoContrast
ImageTransformAutoLevel
ImageTransformBlur
ImageTransformBlurParameter
ImageTransformBrightness
ImageTransformContrast
ImageTransformCrop
ImageTransformCropMargin
ImageTransformDesaturate
ImageTransformDifferenceEast
ImageTransformDifferenceNorth
ImageTransformDifferenceNorthEast
ImageTransformDifferenceNorthWest
ImageTransformDifferenceSouth
ImageTransformDifferenceSouthEast
ImageTransformDifferenceSouthWest
ImageTransformDifferenceWest
ImageTransformDiffuse
ImageTransformEqualize
ImageTransformFlipHorizontally
ImageTransformFlipVertically
ImageTransformGamma
ImageTransformGaussianBlur
ImageTransformGrayscale
ImageTransformHighPass1
ImageTransformHighPass2
ImageTransformHighPass3
ImageTransformInvert
ImageTransformInvertAt
ImageTransformLaplace1
ImageTransformLaplace2
ImageTransformLowPass1
ImageTransformLowPass2
ImageTransformLowPass3
ImageTransformMedianCross
ImageTransformMedianSquare
ImageTransformMedianSquare5
ImageTransformMotionBlurDiagonal1
ImageTransformMotionBlurDiagonal2
ImageTransformMotionBlurHorizontal
ImageTransformMotionBlurVertical
ImageTransformPosterize
ImageTransformRotate
ImageTransformSharpen
ImageTransformSharpenMore
ImageTransformSharpenParameter
ImageTransformThreshold
ImageTransformThresholdBlack
ImageTransformThresholdWhite
ImageTransformTile
ImageTransformTileMedian
ImageUnlink
ItemsSelectAll
ItemsSelectInverse
ItemsSelectNone
LabelsLabelBranchDelete
LabelsLabelBranchDuplicate
LabelsLabelBranchSnap
LabelsLabelBranchSplit
LabelsLabelCenter
LabelsLabelCoordinateAdd
LabelsLabelCoordinateAddMidSegment
LabelsLabelCoordinateAddOnSegment
LabelsLabelCoordinateDelete
LabelsLabelCoordinateDeleteSplitBranch
LabelsLabelCoordinateDuplicate
LabelsLabelCoordinateSnap
LabelsLabelCoordinates
LabelsLabelCoordinatesAddBranch
LabelsLabelCoordinatesAddPoint
LabelsLabelCoordinatesBreakBranch
LabelsLabelCoordinatesDelete
LabelsLabelCoordinatesDeleteBranch
LabelsLabelCoordinatesMoveDown
LabelsLabelCoordinatesMoveToBottom
LabelsLabelCoordinatesMoveToTop
LabelsLabelCoordinatesMoveUp
LabelsLabelCoordinatesNative
LabelsLabelCopy
LabelsLabelCut
LabelsLabelDelete
LabelsLabelDuplicate
LabelsLabelEdit
LabelsLabelOrthogonalize
LabelsLabelPaste
LabelsLabelPasteAppend
LabelsLabelSegmentDelete
LabelsLabelSegmentDeleteSplitBranch
LabelsLabelSegmentize
LabelsLabelSimplify
LabelsLabelSnap
LabelsOpenDrawing
LabelsOpenTable
LabelsOrthogonalize
LabelsPerLabelFormat
LabelsSegmentize
LabelsSimplify
LabelsSnap
LabelsSynchronize
LabelsText
LabelsUnlink
LayoutElementBestFit
LayoutElementColumnsMoveDown
LayoutElementColumnsMoveToBottom
LayoutElementColumnsMoveToTop
LayoutElementColumnsMoveUp
LayoutElementColumnsSelectAll
LayoutElementColumnsSelectInverse
LayoutElementColumnsSelectNone
LayoutElementCopy
LayoutElementCut
LayoutElementDelete
LayoutElementEdit
LayoutElementMoveDown
LayoutElementMoveToBottom
LayoutElementMoveToTop
LayoutElementMoveUp
LayoutElementOpen
LayoutElementOpenInNewWindow
LayoutElementPaste
LayoutElementPasteAppend
LayoutElementProperties
LayoutHide
LayoutLockCenterScale
LayoutLockRectangle
LayoutOpen
LayoutPages
LayoutPrint
LayoutProperties
LayoutTemplateApply
LayoutTemplateApplyFile
LayoutTemplateSave
LayoutZoom
MapAddLayers
MapAddLayersMoveDown
MapAddLayersMoveToBottom
MapAddLayersMoveToTop
MapAddLayersMoveUp
MapAddLayersSelectAll
MapAddLayersSelectInverse
MapAddLayersSelectNone
MapAddNewDrawing
MapAddNewImage
MapAddNewLabels
MapLayerAddLayers
MapLayerAddNewDrawing
MapLayerAddNewImage
MapLayerAddNewLabels
MapLayerCenter
MapLayerCopy
MapLayerCut
MapLayerDelete
MapLayerDuplicate
MapLayerMatch
MapLayerMoveDown
MapLayerMoveToBottom
MapLayerMoveToTop
MapLayerMoveUp
MapLayerOpacity
MapLayerOpen
MapLayerOpenInNewWindow
MapLayerPaste
MapLayerPasteAppend
MapLayerProjectToMap
MapLayerProperties
MapLayerRegister
MapLayerRename
MapLayerRestrictions
MapLayerTransferHeights
MapLayerTransferSelection
MapLayerUseProjection
MapLayerVisible
MapLayerVisibleArea
MapLayerZoom
MapLayerZoomToSelection
MapMatch
MapSpatialOverlay
MapTopologyOverlay
MapTransferHeights
MapTransferSelection
MapTransferSelectionSelectAll
MapTransferSelectionSelectInverse
MapTransferSelectionSelectNone
MapVisibleArea
MouseCenterPoint
MouseDefault
MouseEditBranchDelete
MouseEditBranchSplit
MouseEditCoordinateAdd
MouseEditCoordinateAddOnSegment
MouseEditCoordinateDelete

MouseEditCoordinateDeleteSplitBranch
MouseEditExtend
MouseEditLeave
MouseEditSplit
MouseEditTrim
MouseGrabber
MouseInsertActiveX
MouseInsertAnimation
MouseInsertArea
MouseInsertBox
MouseInsertBoxOnCenter
MouseInsertChart
MouseInsertCheckBox
MouseInsertCircle
MouseInsertCircleGeographic
MouseInsertCircleOnCenter
MouseInsertComboBox
MouseInsertCommandButton
MouseInsertCommonDialog
MouseInsertCoolBar
MouseInsertDateTimePicker
MouseInsertEllipse
MouseInsertEllipseOnCenter
MouseInsertFrame
MouseInsertFreeform
MouseInsertHorizontalScrollBar
MouseInsertImage
MouseInsertImageComboBox
MouseInsertImageList
MouseInsertLabel
MouseInsertLabelLine
MouseInsertLabelLineFreeform
MouseInsertLegend
MouseInsertLine
MouseInsertLineFreeform
MouseInsertLineHorizontal
MouseInsertLineSequence
MouseInsertLineVertical
MouseInsertListBox
MouseInsertListView
MouseInsertMaskedTextBox
MouseInsertMonthView
MouseInsertMultimedia
MouseInsertNorthArrow
MouseInsertOleObject
MouseInsertOptionButton
MouseInsertPictureClip
MouseInsertPoint
MouseInsertProgressBar
MouseInsertRichTextBox
MouseInsertScaleBar
MouseInsertSlider
MouseInsertStaticText
MouseInsertStatusBar
MouseInsertSystemInfo
MouseInsertTabStrip
MouseInsertText
MouseInsertTextBox
MouseInsertToolBar
MouseInsertTreeView
MouseInsertUpDown
MouseInsertVerticalScrollBar
MousePaintAirBrush
MousePaintArea
MousePaintAreaFreeform
MousePaintBox
MousePaintBoxOnCenter
MousePaintBrush
MousePaintBucket
MousePaintCircle
MousePaintCircleOnCenter
MousePaintEllipse
MousePaintEllipseOnCenter
MousePaintEraser
MousePaintGradient
MousePaintLine
MousePaintText
MousePickColor
MouseSelectArea
MouseSelectAreaFreeform
MouseSelectBox
MouseSelectBoxOnCenter
MouseSelectCircle
MouseSelectCircleOnCenter
MouseSelectEllipse
MouseSelectEllipseOnCenter
MouseSelectTouch
MouseTraceArea
MouseTraceAuto
MouseTraceLine
MouseTracePoint
MouseTracker
MouseZoomBox
MouseZoomIn
MouseZoomOut
PaletteOpenImage
ProfileCoordinateAdd
ProfileCoordinateDelete
ProfileCoordinateDuplicate
ProfileCoordinates
ProfileCoordinatesAdd
ProfileCoordinatesDelete
ProfileCoordinatesMoveDown
ProfileCoordinatesMoveToBottom
ProfileCoordinatesMoveToTop
ProfileCoordinatesMoveUp
ProfileCoordinatesNative
ProfileCoordinatesPrint
ProfileOpenSurface
QueryOpenTable
QueryRun
ScriptCompileToDll
ScriptLanguage
ScriptOpenForm
ScriptOpenTable
ScriptPause
ScriptReferences
ScriptReferencesAdd
ScriptReferencesDelete
ScriptReferencesMoveDown
ScriptReferencesMoveToBottom
ScriptReferencesMoveToTop
ScriptReferencesMoveUp
ScriptRun
ScriptRunUnderDebugger
ScriptStepInto
ScriptStepOut
ScriptStepOver
ScriptStop
SurfaceContours
SurfaceContoursAdd
SurfaceContoursAddSequence
SurfaceContoursDelete
SurfaceContoursModeDown
SurfaceContoursMoveToBottom
SurfaceContoursMoveToTop
SurfaceContoursMoveUp
SurfaceConvertTo
SurfaceFilter
SurfaceInvert
SurfaceNoise
SurfaceOpenDataSource
SurfacePosterize
SurfaceQuantize
SurfaceRelink
SurfaceResize
SurfaceThreshold
SurfaceTile
SurfaceTransform
SurfaceTransformAddMargin
SurfaceTransformBlur
SurfaceTransformBlurParameter
SurfaceTransformCrop
SurfaceTransformCropMargin
SurfaceTransformDifferenceEast
SurfaceTransformDifferenceNorth
SurfaceTransformDifferenceNorthEast
SurfaceTransformDifferenceNorthWest
SurfaceTransformDifferenceSouth
SurfaceTransformDifferenceSouthEast
SurfaceTransformDifferenceSouthWest
SurfaceTransformDifferenceWest
SurfaceTransformFlipHorizontally
SurfaceTransformFlipVertically
SurfaceTransformHighPass1
SurfaceTransformHighPass2
SurfaceTransformHighPass3
SurfaceTransformInterpolate
SurfaceTransformInterpolateParameter
SurfaceTransformInterpolateRow
SurfaceTransformInvert
SurfaceTransformLaplace1
SurfaceTransformLaplace2
SurfaceTransformLowPass1
SurfaceTransformLowPass2
SurfaceTransformLowPass3
SurfaceTransformMedianCross
SurfaceTransformMedianSquare
SurfaceTransformMedianSquare5
SurfaceTransformRotate
SurfaceTransformSharpen
SurfaceTransformSharpenMore
SurfaceTransformSharpenParameter
SurfaceTransformThresholdLower
SurfaceTransformThresholdUpper
SurfaceTransformTile
SurfaceTransformTileMedian
SurfaceUnlink
SurfaceWatersheds
TableBestFit
TableBestFitAll
TableBestFitTitles
TableColumnAdd
TableColumnAddRank
TableColumnAddRankAdd
TableColumnAddRankDelete
TableColumnAddRankProperties
TableColumnAddRankShowTypes

TableColumnAddScript
TableColumnChangeType
TableColumnCopy
TableColumnCut
TableColumnDelete
TableColumnEdit
TableColumnEditScript
TableColumnFind
TableColumnFlatten
TableColumnFlattenAll
TableColumnFormat
TableColumnHide
TableColumnIdentity
TableColumnLanguage
TableColumnMoveDown
TableColumnMoveToBottom
TableColumnMoveToTop
TableColumnMoveUp
TableColumnPaste
TableColumnPasteAppend
TableColumnRecompute
TableColumnRename
TableColumnReplace
TableColumnSortAscending
TableColumnSortDescending
TableColumnTransferRules
TableColumnWidth
TableDesign
TableDesignAdd
TableDesignDelete
TableDesignEditValues
TableDesignMoveDown
TableDesignMoveToBottom
TableDesignMoveToTop
TableDesignMoveUp
TableDesignShowDefaultValues
TableDesignShowExtendedProperties
TableDesignShowLanguages
TableDesignShowTransferRules
TableDesignShowTypes
TableGeocode
TableMatch
TableMatchMatchAll
TableMatchMoveDown
TableMatchMoveToBottom
TableMatchMoveToTop
TableMatchMoveUp
TableMatchUseCase
TableMatchUseInteriorWhitespace
TableMatchUseSideWhitespace
TableOpenDataSource
TableOpenDrawing
TableOpenScript
TableQueryAreas
TableQueryBottom
TableQueryContaining
TableQueryContainingMatch
TableQueryContainingToken
TableQueryDuplicates
TableQueryDuplicatesExceptFirst
TableQueryEmpty
TableQueryEnding
TableQueryEndingWithMatch
TableQueryEndingWithToken
TableQueryEqual
TableQueryGreater
TableQueryGreaterOrEqual
TableQueryLess
TableQueryLessOrEqual
TableQueryLines
TableQueryMatching
TableQueryMaximum
TableQueryMedian
TableQueryMinimum
TableQueryNotEqual
TableQueryPoints
TableQuerySizeEqual
TableQuerySizeGreater
TableQuerySizeGreaterOrEqual
TableQuerySizeLess
TableQuerySizeLessOrEqual
TableQuerySizeNotEqual
TableQuerySounding
TableQueryStarting
TableQueryStartingWithMatch
TableQueryStartingWithToken
TableQueryTop
TableQueryTypical
TableQueryUniques
TableRecordAdd
TableRecordCancel
TableRecordClear
TableRecordCoordinates
TableRecordCopy
TableRecordCut
TableRecordDelete
TableRecordFilter
TableRecordMore
TableRecordMoreLikeThis
TableRecordMoreLikeThisSorted
TableRecordPaste
TableRecordPasteAppend
TableRelations
TableRelationsAdd
TableRelationsDelete
TableRelationsIncludeAll
TableRelationsIncludeInverse
TableRelationsIncludeNone
TableRelationsProperties
TableRelink
TableSendEmail
TableStandardize
TableTransformAdd
TableTransformAppend
TableTransformArcCoTangent
TableTransformArcCosine
TableTransformArcSine
TableTransformArcTangent
TableTransformArithmeticSeries
TableTransformBinaryAnd
TableTransformBinaryNot
TableTransformBinaryOr
TableTransformBinaryXor
TableTransformClear
TableTransformCoTangent
TableTransformCopy
TableTransformCopyDay
TableTransformCopyDayName
TableTransformCopyDayOfWeek
TableTransformCopyDayOfYear
TableTransformCopyFirstToken
TableTransformCopyHour
TableTransformCopyLastToken
TableTransformCopyMinute
TableTransformCopyMonth
TableTransformCopyMonthName
TableTransformCopyPercentage
TableTransformCopyRecordDeviation
TableTransformCopyRecordVariation
TableTransformCopyRunningProduct
TableTransformCopyRunningSub
TableTransformCopyRunningTotal
TableTransformCopySecond
TableTransformCopyUrlObject
TableTransformCopyUrlParameters
TableTransformCopyUrlPort
TableTransformCopyUrlProtocol
TableTransformCopyUrlServer
TableTransformCopyWeek
TableTransformCopyYear
TableTransformCopyYearLeapFlag
TableTransformCosine
TableTransformDecimalFraction
TableTransformDecode
TableTransformDecreaseDay
TableTransformDecreaseHour
TableTransformDecreaseMinute
TableTransformDecreaseMonth
TableTransformDecreaseSecond
TableTransformDecreaseWeek
TableTransformDecreaseYear
TableTransformDeleteContaining
TableTransformDeleteContainingMatch
TableTransformDeleteContainingToken
TableTransformDeleteEnding
TableTransformDeleteEndingWithMatch
TableTransformDeleteEndingWithToken
TableTransformDeleteFirstToken
TableTransformDeleteLastToken
TableTransformDeleteLeft
TableTransformDeleteMatching
TableTransformDeleteRight
TableTransformDeleteSounding
TableTransformDeleteStarting
TableTransformDeleteStartingWithMatch
TableTransformDeleteStartingWithToken
TableTransformDivide
TableTransformEncode
TableTransformExponentiate
TableTransformFill
TableTransformFillWithAverage
TableTransformFillWithE
TableTransformFillWithExcess
TableTransformFillWithMaximum
TableTransformFillWithMedian
TableTransformFillWithMinimum
TableTransformFillWithPi
TableTransformFillWithRandom
TableTransformFillWithSkewness
TableTransformFillWithStdDeviation
TableTransformFillWithSum
TableTransformFillWithVariance
TableTransformGeometricSeries
TableTransformIncreaseDay
TableTransformIncreaseHour
TableTransformIncreaseMinute
TableTransformIncreaseMonth
TableTransformIncreaseSecond

TableTransformIncreaseWeek
TableTransformIncreaseYear
TableTransformInvert
TableTransformLeaveContaining
TableTransformLeaveContainingMatch
TableTransformLeaveContainingToken
TableTransformLeaveEnding
TableTransformLeaveEndingWithMatch
TableTransformLeaveEndingWithToken
TableTransformLeaveLeft
TableTransformLeaveMatching
TableTransformLeaveRight
TableTransformLeaveSounding
TableTransformLeaveStarting
TableTransformLeaveStartingWithMatch
TableTransformLeaveStartingWithToken
TableTransformLogarithm
TableTransformLogarithmBinary
TableTransformLogarithmDecimal
TableTransformLogarithmNatural
TableTransformLogicalAnd
TableTransformLogicalNot
TableTransformLogicalOr
TableTransformLogicalXor
TableTransformMakeAbsolute
TableTransformMakeAtLeast
TableTransformMakeAtMost
TableTransformMakeLowerCase
TableTransformMakeSentenceCase
TableTransformMakeTitleCase
TableTransformMakeUpperCase
TableTransformModulo
TableTransformMultiply
TableTransformNegate
TableTransformPower
TableTransformPrepend
TableTransformReverse
TableTransformReverseByTokens
TableTransformReverseToken
TableTransformRound
TableTransformRoundDown
TableTransformRoundUp
TableTransformSine
TableTransformSpanExcluding
TableTransformSpanIncluding
TableTransformSquare
TableTransformSquareRoot
TableTransformSubtract
TableTransformTangent
TableTransformToggleCase
TableTransformTrim
TableTransformTrimLeft
TableTransformTrimRight
TableUnlink
TableValueAdd
TableValueCancel
TableValueClear
TableValueClearAll
TableValueCopy
TableValueCut
TableValueDelete
TableValueMore
TableValueMoreLikeThis
TableValueMoreLikeThisSorted
TableValuePaste
TableValuePasteAppend
TableValueRunUnderDebugger
TableViewBotAreas
TableViewBotAverage
TableViewBotBottom
TableViewBotCenterMoment
TableViewBotContaining
TableViewBotContainingMatch
TableViewBotContainingToken
TableViewBotDuplicates
TableViewBotDuplicatesExceptFirst
TableViewBotEmpty
TableViewBotEnding
TableViewBotEndingWithMatch
TableViewBotEndingWithToken
TableViewBotEqual
TableViewBotExcess
TableViewBotGreater
TableViewBotGreaterOrEqual
TableViewBotLess
TableViewBotLessOrEqual
TableViewBotLines
TableViewBotMatching
TableViewBotMaximum
TableViewBotMedian
TableViewBotMinimum
TableViewBotMoment
TableViewBotNotEqual
TableViewBotPoints
TableViewBotRange
TableViewBotRecords
TableViewBotSizeEqual
TableViewBotSizeGreaterOrEqual
TableViewBotSizeGreter
TableViewBotSizeLess
TableViewBotSizeLessOrEqual
TableViewBotSizeNotEqual
TableViewBotSkew
TableViewBotSounding
TableViewBotStarting
TableViewBotStartingWithMatch
TableViewBotStartingWithToken
TableViewBotStdDeviation
TableViewBotSum
TableViewBotTop
TableViewBotTypical
TableViewBotUniques
TableViewBotVariance
TerrainClouds
TerrainFog
TerrainLighting
TerrainOpenSurface
TerrainOverlay
TerrainOverlayMoveDown
TerrainOverlayMoveToBottom
TerrainOverlayMoveToTop
TerrainOverlayMoveUp
TerrainOverlayShowAll
TerrainOverlayShowInverse
TerrainOverlayShowNone
TerrainSnapToSurface
TerrainSurface
TerrainWater
ThemeOpenDrawing
ThemeOpenTable
ToolsAddinList
ToolsAddins
ToolsAddinsSelectAll
ToolsAddinsSelectInverse
ToolsAddinsSelectNone
ToolsAdministratorConsole
ToolsAdministratorConsoleColumns
ToolsAdministratorConsoleGlobalFilter
ToolsAdministratorConsoleProperties
ToolsAdministratorConsoleRefresh
ToolsAdministratorConsoleRun
ToolsAdministratorConsoleTriggers
ToolsBatchExport
ToolsCustomize
ToolsDatabaseConsole
ToolsDatabaseConsoleComponentView
ToolsDatabaseConsoleGlobalFilter
ToolsDatabaseConsoleImport
ToolsDatabaseConsoleImportQuery
ToolsDatabaseConsoleLink
ToolsDatabaseConsoleLinkQuery
ToolsDatabaseConsoleRefresh
ToolsDatabaseConsoleRun
ToolsMakeImage
ToolsOptions
ToolsOptionsPalettesSave
ToolsServerConsole
ToolsServerConsoleAddFolder
ToolsServerConsoleDelete
ToolsServerConsoleImport
ToolsServerConsoleLink
ToolsServerConsoleRefresh
ToolsServerConsoleUndoCheckOut
ViewCallStack
ViewCallStackGoTo
ViewCallStackShowLanguage
ViewCallStackShowLine
ViewCallStackShowLinePosition
ViewCallStackShowModule
ViewColumns
ViewColumnsMoveDown
ViewColumnsMoveToBottom
ViewColumnsMoveToTop
ViewColumnsMoveUp
ViewColumnsShowAll
ViewColumnsShowInverse
ViewColumnsShowNone
ViewControlPoints
ViewControlPointsAdd
ViewControlPointsAddBlank
ViewControlPointsDelete
ViewControlPointsLoad
ViewControlPointsLoadFromFile
ViewControlPointsNative
ViewControlPointsPreview
ViewControlPointsRegister
ViewControlPointsSave
ViewControlPointsSaveToFile
ViewControlPointsShowNames
ViewDisplayOptionsElevation
ViewDisplayOptionsImage
ViewDisplayOptionsLabels
ViewDisplayOptionsProfile
ViewDisplayOptionsSurface
ViewDisplayOptionsSurfaceApply
ViewDisplayOptionsSurfaceDarken
ViewDisplayOptionsSurfaceGrayscale
ViewDisplayOptionsSurfaceInterpolate

ViewDisplayOptionsSurfaceLighten
ViewDisplayOptionsSurfaceLoadFromFile
ViewDisplayOptionsSurfaceReverse
ViewDisplayOptionsSurfaceSaveToFile
ViewErrors
ViewErrorsGoTo
ViewErrorsShowLine
ViewErrorsShowLinePosition
ViewErrorsShowModule
ViewFilterSelection
ViewFullScreen
ViewGoBack
ViewGoForward
ViewGpsConsole
ViewGpsConsoleConnect
ViewGpsConsoleDownloadWaypoints
ViewGpsConsoleProperties
ViewGpsConsoleTrack
ViewGpsConsoleTrackCenter
ViewGpsConsoleTrackColumns
ViewGpsConsoleTrackLine
ViewGpsConsoleTrackPoints
ViewGpsConsoleTrackSequence
ViewGpsConsoleUploadWaypoints
ViewGraticule
ViewGrid
ViewHistory
ViewHistoryClear
ViewHistoryCopy
ViewHistoryFreeze
ViewHistoryPrint
ViewHistorySaveToFile
ViewInfo
ViewInfoCenter
ViewInfoFilterSelection
ViewInfoGoFirst
ViewInfoGoLast
ViewInfoGoNext
ViewInfoGoPrevious
ViewInfoPing
ViewInfoShowBlanks
ViewInfoShowIntrinsics
ViewInfoShowTypes
ViewInfoZoom
ViewLayers
ViewLayersDelete
ViewLayersHideAll
ViewLayersMoveDown
ViewLayersMoveToBottom
ViewLayersMoveToTop
ViewLayersMoveUp
ViewLayersOpacity
ViewLayersProperties
ViewLayersRestrictions
ViewLayersShowAll
ViewLayersSwitchTo
ViewLegend
ViewLegendAlignLeftBottom
ViewLegendAlignLeftTop
ViewLegendAlignNone
ViewLegendAlignRightBottom
ViewLegendAlignRightTop
ViewLegendCopy
ViewLegendCopyStructure
ViewLegendElementAdd
ViewLegendElementAddSamples
ViewLegendElementDelete
ViewLegendElementFlatten
ViewLegendElementMoveDown
ViewLegendElementMoveToBottom
ViewLegendElementMoveToTop
ViewLegendElementMoveUp
ViewLegendHide
ViewLegendPaste
ViewLegendPasteAppend
ViewLegendProperties
ViewNorthArrow
ViewNorthArrowAlignLeftBottom
ViewNorthArrowAlignLeftTop
ViewNorthArrowAlignNone
ViewNorthArrowAlignRightBottom
ViewNorthArrowAlignRightTop
ViewNorthArrowCopy
ViewNorthArrowHide
ViewNorthArrowPaste
ViewNorthArrowProperties
ViewNotes
ViewNotesClear
ViewNotesLoadFromFile
ViewNotesPrint
ViewNotesSaveToFile
ViewProject
ViewProjectAssignProjection
ViewProjectCached
ViewProjectChangeProjection
ViewProjectCheckIn
ViewProjectCheckOut
ViewProjectCopy
ViewProjectCreate
ViewProjectCreateChart
ViewProjectCreateComments
ViewProjectCreateDrawing
ViewProjectCreateElevation
ViewProjectCreateFolder
ViewProjectCreateForm
ViewProjectCreateImage
ViewProjectCreateLabels
ViewProjectCreateLayout
ViewProjectCreateMap
ViewProjectCreatePalette
ViewProjectCreateProfile
ViewProjectCreateQuery
ViewProjectCreateScript
ViewProjectCreateSurface
ViewProjectCreateTable
ViewProjectCreateTerrain
ViewProjectCreateTheme
ViewProjectCreateZones
ViewProjectCut
ViewProjectDelete
ViewProjectDesign
ViewProjectDuplicate
ViewProjectExport
ViewProjectGetLatestVersion
ViewProjectMakeImage
ViewProjectOpen
ViewProjectOpenDataSource
ViewProjectOpenInNewWindow
ViewProjectPaste
ViewProjectPasteAs
ViewProjectPasteAsChart
ViewProjectPasteAsComments
ViewProjectPasteAsDrawing
ViewProjectPasteAsElevation
ViewProjectPasteAsForm
ViewProjectPasteAsImage
ViewProjectPasteAsLabels
ViewProjectPasteAsLayout
ViewProjectPasteAsMap
ViewProjectPasteAsPalette
ViewProjectPasteAsProfile
ViewProjectPasteAsQuery
ViewProjectPasteAsScript
ViewProjectPasteAsSurface
ViewProjectPasteAsTable
ViewProjectPasteAsTerrain
ViewProjectPasteAsTheme
ViewProjectPasteAsZones
ViewProjectPrint
ViewProjectProperties
ViewProjectRefresh
ViewProjectRefreshData
ViewProjectRelink
ViewProjectRename
ViewProjectRun
ViewProjectRunUnderDebugger
ViewProjectShare
ViewProjectUnlink
ViewProjectUnlinkLabels
ViewProjectUnshare
ViewProperties
ViewRefresh
ViewRefreshAuto
ViewRefreshData
ViewReview
ViewReviewCenter
ViewReviewColumns
ViewReviewUseAllLocal
ViewReviewUseAllRemote
ViewReviewUseLocal
ViewReviewUseRemote
ViewReviewViewConflict
ViewReviewZoom
ViewScaleBar
ViewScaleBarAlignLeftBottom
ViewScaleBarAlignLeftTop
ViewScaleBarAlignNone
ViewScaleBarAlignRightBottom
ViewScaleBarAlignRightTop
ViewScaleBarCopy
ViewScaleBarHide
ViewScaleBarPaste
ViewScaleBarProperties
ViewSelections
ViewSelectionsAdd
ViewSelectionsDelete
ViewSelectionsPreview
ViewSelectionsSelectAdd
ViewSelectionsSelectIntersect
ViewSelectionsSelectInvert
ViewSelectionsSelectReplace
ViewSelectionsSelectSubtract
ViewSort
ViewSortMoveDown
ViewSortMoveToBottom
ViewSortMoveToTop
ViewSortMoveUp
ViewSortUseCase

ViewSortUseInteriorWhitespace
ViewSortUseSideWhitespace
ViewStructureBranchBoxes
ViewStructureBranchCentroids
ViewStructureInflectionPoints
ViewStructureObjectBoxes
ViewStructureObjectCentroids
ViewStructureSegmentPoints
ViewStructureTieLines
ViewStructureTiePoints
ViewToolProperties
ViewVariables
ViewVariablesShowCommonProperties
ViewVariablesShowType
ViewViewBots
ViewViewBotsAdd
ViewViewBotsDelete
ViewViewBotsMoveDown
ViewViewBotsMoveToBottom
ViewViewBotsMoveToTop
ViewViewBotsMoveUp
ViewViewBotsRefresh
ViewViewBotsRefreshAll
ViewViewBotsSelectAdd
ViewViewBotsSelectIntersect
ViewViewBotsSelectInvert
ViewViewBotsSelectReplace
ViewViewBotsSelectSubtract
ViewViewBotsSequence
ViewViewBotsSequenceArgument
ViewViewBotsSequenceColumn
ViewViewBotsSequenceOperation
ViewViewBotsSequenceScope
ViewViewBotsSequenceScopeSize
ViewViewBotsSequenceTime
ViewViewBotsSequenceValue
ViewViews
ViewViewsAdd
ViewViewsApply
ViewViewsDelete
ViewWatches
ViewWatchesAdd
ViewWatchesDelete
ViewWatchesShowCommonProperties
ViewWatchesShowType
ViewWorld
ViewWorldShowPosition
ViewWorldTrackPosition
ViewZoom
ViewZoomIn
ViewZoomNative
ViewZoomOut
ViewZoomTo
ViewZoomToFit
WindowCascade
WindowClose
WindowCloseAll
WindowNew
WindowNext
WindowPrevious
WindowTileHorizontally
WindowTileVertically
WindowWindows
WindowWindowsActivate
WindowWindowsCascade
WindowWindowsClose
WindowWindowsTileHorizontally
WindowWindowsTileVertically

Preset Definintions: Distance and Area Units

Extracted from xml at:
http://www.manifold.net/products/mfd50pro/Units.xml

Name	Nickname	Nickname Area	Value	Lat/Long
Arc Minute	arcmin	sq arcmin	1.66666666666666600000000e-002	True
Arc Second	arcsec	sq arcsec	2.77777777777777800000000e-004	True
Arshin	arsh	sq arsh	7.11200000000000500000000e-001	
Benoit Chain A	ch Ben A	sq ch Ben A	2.01167824000000020000000e+001	
Benoit Chain B	ch Ben B	sq ch Ben B	2.01167824943759010000000e+001	
Benoit Foot A	ft Ben A	sq ft Ben A	3.04799733333329900000000e-001	
Benoit Foot B	ft Ben B	sq ft Ben B	3.04799734763270990000000e-001	
Benoit Link A	ln Ben A	sq ln Ben A	2.01167824000000000000000e-001	
Benoit Link B	ln Ben B	sq ln Ben B	2.01167824943759010000000e-001	
Benoit Yard A	yd Ben A	sq yd Ben A	9.14399199999999700000000e-001	
Benoit Yard B	yd Ben B	sq yd Ben B	9.14399204289812030000000e-001	
Cable	ca	sq ca	1.85312999999999900000000e+002	
Centimeter	cm	sq cm	1.00000000000000000000000e-002	
Chain	ch	sq ch	2.01166194999999990000000e+001	
Clarke's Chain	ch Cla	sq ch Cla	2.01166194900000010000000e+001	
Clarke's Foot	ft Cla	sq ft Cla	3.04797265099999990000000e-001	
Clarke's Link	ln Cla	sq ln Cla	2.01166194900000010000000e-001	
Clarke's Yard	yd Cla	sq yd Cla	9.14391795000000010000000e-001	
Decimeter	dm	sq dm	1.00000000000000010000000e-001	
Fathom	f	sq f	1.82880000000000000000000e+000	
Foot	ft	sq ft	3.04800000000000200000000e-001	
German Legal Meter	Glm	sq Glm	1.00001359650000010000000e+000	
Inch	in	sq in	2.53999999999999900000000e-002	
Indian Foot	Ind ft	sq Ind ft	3.04787605800000300000000e-001	
Indian Foot (1937)	Ind ft 37	sq Ind ft 37	3.04798410000000200000000e-001	
Indian Foot (1962)	Ind ft 62	sq Ind ft 62	3.04799600000000000000000e-001	
Indian Foot (1975)	Ind ft 75	sq Ind ft 75	3.04799500000000000000000e-001	
Indian Yard	Ind yd	sq Ind yd	9.14398530744441000000000e-001	
Indian Yard (1937)	Ind yd 37	sq Ind yd 37	9.14395230000000000000000e-001	
Indian Yard (1962)	Ind yd 62	sq Ind yd 62	9.14398799999999600000000e-001	
Indian Yard (1975)	Ind yd 75	sq Ind yd 75	9.14398500000000000000000e-001	
Kilometer	km	sq km	1.00000000000000000000000e+003	
Kosaya Sazhen	kszh	sq kszh	2.48000000000000000000000e+000	
League	lg	sq lg	4.82789999999999600000000e+003	
Link	ln	sq ln	2.01166194999999990000000e-001	
Micrometer	mcm	sq mcm	1.00000000000000000000000e-004	
Mile	mi	sq mi	1.60934400000000100000000e+003	
Millimeter	mm	sq mm	1.00000000000000000000000e-003	
Nautical Mile	nm	sq nm	1.85200000000000000000000e+003	
Radian	rad	sq rad	5.72957795128959970000000e+001	True
Rod	r	sq r	5.02920000000000300000000e+000	
Sazhen	szh	sq szh	2.13330000000000200000000e+000	
Sea League	slg	sq slg	5.55600000000000000000000e+003	
Sears Chain	ch Srs	sq ch Srs	2.01167651215526000000000e+001	
Sears Foot	ft Srs	sq ft Srs	3.04799471538675980000000e-001	
Sears Link	ln Srs	sq ln Srs	2.01167651215525990000000e-001	
Sears Yard	yd Srs	sq yd Srs	9.14398414616029000000000e-001	

Name	Nickname	Nickname Area	Value	Lat/Long
US Survey Chain	ch US	sq ch US	2.01168402336804990000000e+001	
US Survey Foot	ft US	sq ft US	3.04800609599999990000000e-001	
US Survey Link	ln US	sq ln US	2.01168402336805000000000e-001	
US Survey Mile	mi US	sq mi US	1.60934721869444000000000e+003	
Vershok	Vh	sq vh	4.44500000000000300000000e-002	
Verst	Ve	sq ve	1.06680000000000000000000e+003	
Yard	Yd	sq yd	9.14400000000001000000000e-001	

Preset Definitions: Style Lists (System Dependant)

Manifold System 7x Release Notes
http://www.manifold.net/updates/release_notes_7x.htm
The command line has been extended with a new "slist" option, which dumps the names of all formatting styles available for areas, labels, lines and points into a text file.

Example syntax: "manifold.exe /slist:file.txt".

To specify the full program path and path to the resultant text file, use the following syntax:
"C:\Program Files\Manifold System\manifold.exe" /slist:"c:\temp\Manifold_Formatting_Options.txt"

Table: V7 Area styles (Aug 12, 2006):

brick	doubleV	singleF
brickArab	netBulb	singleG
brickStar	netBulbCross	singleH
checkers	netBulbDeco	singleV
circleDots	netBulbF	solid
circleDotsH	netBulbG	squares
circleDotsQ	netBulbH	stairDown
circles	netBulbV	stairUp
corners	netBulbXCross	stripH
cross	netCross	strips
dashBounds	netCrossSmall	stripThickF
dashDots	netDash	stripThickG
dashH	netH	stripThickH
dashShortH	netV	stripThickV
dashShortV	plain	stripV
dashSwamp	plate	ticksH
dashSwampAlt	plateCombo	tShapes
dashV	plateConcrete	tShapesRev
doubleF	plateConcreteHole	xCheckers
doubleG	plateWrap	xCross
doubleH	plateX	zigzag

Table: V7 Line styles (Aug 12, 2006):

arrow	arrowSpacedOver	capArrow
arrowAlt	arrowSpacedOverRev	capArrowDeco
arrowAltOver	axial	capArrowDecoDot
arrowOutline	axialBold	capArrowDecoRev
arrowOutlineRev	box	capArrowDecoRevDot
arrowOver	boxCenter	capArrowDot
arrowOverRev	boxDot	capArrowLong
arrowRev	boxDotDotDot	capArrowLongDot
arrowSpacedOutlineOver	boxOver	capArrowLongRev
arrowSpacedOutlineOverRev	boxProper	capArrowLongRevDot

capArrowRev
capArrowRevDot
capCircle
capCircleDot
capCircleRev
capCircleRevDot
capDiamond
capDiamondDot
capDiamondRev
capDiamondRevDot
capSquare
capSquareDot
capSquareRev
capSquareRevDot
circle
circleBorder
circleCenter
circleDot
circleDotDotDot
circleOver
Cold Front
cross
crossProper
dash3
dash3Wide
dash4
dash4Wide
dash5
dash5Dot
dash5DotDot
dash5DotDotDot
dash5Wide
dash6
dash7
dash8
dash9
diamond
diamondCenter
diamondDot
diamondDotDotDot
diamondOver
diamondProper
dot
dotWide
dotWideX
dotWideXX
double
Occulsion
railway
railwayHollow
railwayLong
sideArrow
sideArrowOver
sideArrowOverRev
sideArrowOverSymm
sideArrowRev
sideArrowSymm
sideBump
sideBumpRev
sideBumpSymm
sideTick
sideTickRev
sideTickSymm
solid
ticks
ticksOver
ticksPro
ticksWall
triangleProper
triangleRevProper
Trowal
Warm Front
wave
xCross
xCrossOver
xCrossProper

Table: V7 Point styles (Aug 12, 2006):

arc
arcClosed
arrowB
arrowL
arrowLB
arrowLT
arrowR
arrowRB
arrowRT
arrowT
box
boxBDark
boxCross
boxCrossX
boxDot
boxDouble
boxF
boxFDark
boxFDarkRev
boxFull
boxG
boxGDark
boxGDarkRev
boxH
boxLDark
boxRDark
boxSpot
boxTDark
boxV
brick
centimo
chart
chart12
chart25
chart37
chart50
chart62
chart75
chart87
circle
circleBDark
circleCity
circleCityX
circleCross
circleCrossX
circleDot
circleDouble
circleF
circleFDark
circleFDarkRev
circleFull

circleG
circleGDark
circleGDarkRev
circleH
circleLDark
circleRDark
circleSpot
circleTDark
circleV
colon
cone
coneDark
cross
crossH
crossS
crossX
crossXM
cruzeiro
diamond
diamondBDark
diamondCross
diamondCrossX
diamondDot
diamondDouble
diamondF
diamondFDark
diamondFDarkRev
diamondFull
diamondG
diamondGDark
diamondGDarkRev
diamondH
diamondLDark
diamondRDark
diamondSpot
diamondTDark
diamondV
digit0
digit1
digit2
digit3
digit4
digit5
digit6
digit7
digit8
digit9
dollar
dong
drachma
euro
flagRect
flagRectWipe
flagTriangle
frank
guideB
guideL
guideLB
guideLT
guideR
guideRB
guideRT
guideT
kip
letterA
letterB
letterC
letterD
letterE
letterF
letterG
letterH
letterI
letterJ
letterK
letterL
letterM
letterN
letterO
letterP
letterQ
letterR
letterS
letterT
letterU
letterV
letterW
letterX
letterY
letterZ
lira
mill
naira
peseta
pound
rupee
sheqel
tree
triangle
triangleDot
triangleDouble
triangleFull
triangleLDark
triangleRDark
triangleRev
triangleRevDot
triangleRevDouble
triangleRevFull
triangleRevLDark
triangleRevRDark
triangleRevSpot
triangleRevV
triangleSpot
triangleV
tShape
tShapeRev
tugrik
value
wind
Wingdings 033
Wingdings 034
Wingdings 035
Wingdings 036
Wingdings 037
Wingdings 038
Wingdings 039
Wingdings 040
Wingdings 041
Wingdings 042
Wingdings 043
Wingdings 044
Wingdings 045
Wingdings 046
Wingdings 047
Wingdings 048
Wingdings 049
Wingdings 050
Wingdings 051

Wingdings 052
Wingdings 053
Wingdings 054
Wingdings 055
Wingdings 056
Wingdings 057
Wingdings 058
Wingdings 059
Wingdings 060
Wingdings 061
Wingdings 062
Wingdings 063
Wingdings 064
Wingdings 065
Wingdings 066
Wingdings 067
Wingdings 068
Wingdings 069
Wingdings 070
Wingdings 071
Wingdings 072
Wingdings 073
Wingdings 074
Wingdings 075
Wingdings 076
Wingdings 077
Wingdings 078
Wingdings 079
Wingdings 080
Wingdings 081
Wingdings 082
Wingdings 083
Wingdings 084
Wingdings 085
Wingdings 086
Wingdings 087
Wingdings 088
Wingdings 089
Wingdings 090
Wingdings 091
Wingdings 092
Wingdings 093
Wingdings 094
Wingdings 095
Wingdings 096
Wingdings 097
Wingdings 098
Wingdings 099
Wingdings 100
Wingdings 101
Wingdings 102
Wingdings 103
Wingdings 104
Wingdings 105
Wingdings 106
Wingdings 107
Wingdings 108
Wingdings 109
Wingdings 110
Wingdings 111
Wingdings 112
Wingdings 113
Wingdings 114
Wingdings 115
Wingdings 116
Wingdings 117
Wingdings 118
Wingdings 119
Wingdings 120
Wingdings 121
Wingdings 122
Wingdings 123
Wingdings 124
Wingdings 125
Wingdings 126
Wingdings 127
Wingdings 128
Wingdings 129
Wingdings 130
Wingdings 131
Wingdings 132
Wingdings 133
Wingdings 134
Wingdings 135
Wingdings 136
Wingdings 137
Wingdings 138
Wingdings 139
Wingdings 140
Wingdings 141
Wingdings 142
Wingdings 143
Wingdings 144
Wingdings 145
Wingdings 146
Wingdings 147
Wingdings 148
Wingdings 149
Wingdings 150
Wingdings 151
Wingdings 152
Wingdings 153
Wingdings 154
Wingdings 155
Wingdings 156
Wingdings 157
Wingdings 158
Wingdings 159
Wingdings 160
Wingdings 161
Wingdings 162
Wingdings 163
Wingdings 164
Wingdings 165
Wingdings 166
Wingdings 167
Wingdings 168
Wingdings 169
Wingdings 170
Wingdings 171
Wingdings 172
Wingdings 173
Wingdings 174
Wingdings 175
Wingdings 176
Wingdings 177
Wingdings 178
Wingdings 179
Wingdings 180
Wingdings 181
Wingdings 182
Wingdings 183
Wingdings 184
Wingdings 185
Wingdings 186
Wingdings 187
Wingdings 188
Wingdings 189

Wingdings 190
Wingdings 191
Wingdings 192
Wingdings 193
Wingdings 194
Wingdings 195
Wingdings 196
Wingdings 197
Wingdings 198
Wingdings 199
Wingdings 200
Wingdings 201
Wingdings 202
Wingdings 203
Wingdings 204
Wingdings 205
Wingdings 206
Wingdings 207
Wingdings 208
Wingdings 209
Wingdings 210
Wingdings 211
Wingdings 212
Wingdings 213
Wingdings 214
Wingdings 215
Wingdings 216
Wingdings 217
Wingdings 218
Wingdings 219
Wingdings 220
Wingdings 221
Wingdings 222
Wingdings 223
Wingdings 224
Wingdings 225
Wingdings 226
Wingdings 227
Wingdings 228
Wingdings 229
Wingdings 230
Wingdings 231
Wingdings 232
Wingdings 233
Wingdings 234
Wingdings 235
Wingdings 236
Wingdings 237
Wingdings 238
Wingdings 239
Wingdings 240
Wingdings 241
Wingdings 242
Wingdings 243
Wingdings 244
Wingdings 245
Wingdings 246
Wingdings 247
Wingdings 248
Wingdings 249
Wingdings 250
Wingdings 251
Wingdings 252
Wingdings 253
Wingdings 254
Wingdings 255

Table: Label styles (Aug 12, 2006):

box
boxB
boxCorners
boxFrame
boxGradHorzDD
boxGradHorzDL
boxGradHorzLD
boxGradHorzLL
boxGradVertDD
boxGradVertDL
boxGradVertLD
boxGradVertLL
boxInner
boxL
boxR
boxT
boxThick
boxThickB
boxThickL
boxThickR
boxThickT
halo
haloLB
haloLT
haloRB
haloRT
haloWide
plain
roadShield
roadShieldBlack
roadShieldSmall
roadShieldSmallBlack
roadShieldSmallWhite
roadShieldSmallWire
roadShieldWhite
roadShieldWire

Object Model – V7 Features and Changes

Deselect Records – New V7 SelectNone, SelectAll & SelectInverse Methods

http://69.17.46.171/Site/Thread.aspx?id=21452&ti=632838853471300000

howells at 5/20/2006 5:04 PM (#21453)

I would like to deselect all records (ie select none) through code
Steven

KlausDE at 5/21/2006 4:55 AM (#21459)

coding in VBScript you would use

```
-- VBScript

Sub Main
  Set doc = Application.ActiveDocument
  Set comps = doc.ComponentSet
  Set theQuery = doc.NewQuery("TempQuery", False)
  theQuery.Text = "Update [Cities] SET [Selection (I)] = False;"
  theQuery.RunEx(True)
  comps.Remove(comps.ItemByID(theQuery.ID))
End Sub
```

willh at 5/21/2006 9:53 AM (#21464)

Actually, when using v7 there is a SelectNone method for tables which operates signifcantly faster than the sql method--not to mention how much easier it is to code. **SelectNone** as well as **SelectAll** and **SelectInverse** are available for Drawings, Tables, Surfaces and Images. I've replaced the query method with SelectNone on a surface in an addin; it would take the query over 2 minutes to run while the SelectNone takes about 5 seconds.

willh at 5/21/2006 10:05 AM (#21465)

Also, I'd like to remind everyone to state explicitly what version of Manifold you're using.

And since **these new methods aren't in the documentation**: if you don't have access to or prefer not to use VS for object model browsing, then check out the wonderful .NET Reflector

http://www.aisto.com/roeder/dotnet/

"Document" is a Predefined Keyword – ActiveDocument Often not Required

http://69.17.46.171/Site/Thread.aspx?id=21589&ti=632840185176300000

aforest at 5/23/2006 5:15 PM (#21590)

Why is it that when in Manifold 7 when i

```
Set document = Application.ActiveDocument
```

I get an error " Object doesn't support this method or property "document" but in manifold 6.5 it works perfectly find. If i change the variable "document" to something else like "doc" it works find but if I don't change it i have to declare the variable "document" first then only it works. Can anyone please explain? Thanks.

adamw at 5/23/2006 10:01 PM (#21596)

When Manifold starts a script written in a non-.NET language, such as VBScript, JScript, or, say, Python (meaning ActiveState Python), it creates a set of predefined objects which the script can use to access the current execution context. Thus, 7.00, 6.50 and all other versions of Manifold are creating the Application object, which is what you use when you do:

```
'VBScript
Set doc = Application.ActiveDocument
```

7.00 is also creating the Document object. The Document object points to the active document if you are running an interactive session of Manifold and to the document containing the script if you are using Manifold in the context of a web site or external application.

Adding a new predefined object is always a breaking change and requires making adjustments to existing scripts that use the name of the new object. In this case, we think the change is worth the hassle since it allows script writers to write scripts that can run seamlessly both in interactive and programmatic sessions of Manifold.

We recommend that you always use the Document object, and only resort to Application.ActiveDocument when you are sure this is what you need, since it only works for interactive sessions of Manifold. Hence, it would perhaps be best to fix your case:

```
'VBScript
Set document = Application.ActiveDocument
```

...by removing this line or commenting it out.

Accessing the Records.LastAdded Object

http://69.17.46.171/Site/Thread.aspx?id=21549&ti=632841335964730000

adamw at 5/24/2006 6:32 AM (#21610)

As suggested by Will, change the following code in "cmdGeocode_Click":

```
'VBScript
Do While Not rs.eof
  ...
  objects.Add Application.NewGeom(GeomPoint, point)
  Set record = records.LastAdded
  ...
Loop
```

to:

```
'VBScript
Do While Not rs.eof
  ...
  objects.Add Application.NewGeom(GeomPoint, point)
  Set record = objects.LastAdded.Record
  ...
Loop
```

Now, to clarify what has been mentioned by Art:

In 6.50, the LastAdded property was sometimes bound to a physical object manipulated by the relevant script object (eg, a drawing or a table), and sometimes to the script object itself (eg, an ObjectSet or a RecordSet). When the property was bound to a physical object, such as when adding objects to a drawing, it was possible to "add" data to one script object (an ObjectSet) and retrieve added data using another script object (a RecordSet). This was making the implementation of the script objects that had to reconcile changes with each other unnecessarily complex and slow, and was making it very difficult to use the LastAdded property in multi-threaded scenarios. So...

In 7.00, the LastAdded property is always bound to the script object exposing it. That is, if you add a record to a RecordSet object, you can only access the added record using the exact same RecordSet object you used to add the record. If you create a new RecordSet object, it will not return the added record in the LastAdded property, even if bound to the exact same physical table component. Similarly, if you add an object to an ObjectSet object, you can not retrieve the record for that object using the LastAdded property of the relevant RecordSet object. You should use the LastAdded property of the ObjectSet object to retrieve the added object and locate the record for that object using other means.

Let me know if this sounds Chinese...

Dumping V7X Formatting Options – Style Lists

Manifold System 7x Release Notes
http://www.manifold.net/updates/release_notes_7x.htm
The command line has been extended with a new "slist" option, which dumps the names of all formatting styles available for areas, labels, lines and points into a text file.

Example syntax: "manifold.exe /slist:file.txt".

To specify the full program path and path to the resultant text file, use the following syntax:
"C:\Program Files\Manifold System\manifold.exe" /slist:"c:\temp\Manifold_Formatting_Options.txt"

Objects

IDs Assigned to New Drawing Objects are not Guaranteed to be Sequential

Manifold-L Jan 11, 2006
Author: Adam Wachowski
Date: 01-10-2006 23:55
http://lists.directionsmag.com/discussion/read.php?f=29&i=41307&t=41307

```
> A programmer I am working with wants to resort the
> order of the objects in a drawing exported as midmif.
> ... is it possible for the native sort order of objects to
> be reset and ordered?
```

No. However, as it has been suggested by Mike Sumner, you can re-create the drawing adding the objects in the order you want. The easiest way to do this is to create a query like:

```
SELECT * FROM [Drawing] ORDER BY [DesiredColumn];
```

...and import that as a drawing, using [Geom (I)] as a geometry column.

The most interesting part of this is that while most exports, including the MIF / MID export will export the resulting drawing in the literal order of records in the producing query, the internal specs place no restrictions on

a) the order with which a drawing import processes query records, or
b) the order with which a drawing export processes drawing objects. Given that
c) the IDs assigned to new drawing objects are not guaranteed to be sequential, the above solution is good as a one-off temporary hack but is too fragile for the long term. If you or your friend really want Manifold to export objects to a MIF / MID pair in a specific order, send a suggestion to sales@manifold.net.

Adam Wachowski

Adding/Editing Objects in a Drawing Layer

Manifold©-L
Author: Adam Wachowski
Date: 01-23-2006 08:41
http://lists.directionsmag.com/discussion/read.php?f=29&i=41380&t=41370

Adding an Object

```
Set drw = ... ' your drawing
Set pt = Application.NewPoint(5, 6)
Set geom = Application.NewGeom(GeomPoint, pt)
drw.ObjectSet.Add geom
```

Alter an Object
Alter the Geom property of the relevant Object.

How to Add Objects (Points) to a Drawing

http://69.17.46.171/Site/Thread.aspx?id=18896&ti=632751025401430000

aforest at 2/9/2006 12:00 AM (#18897)
Can anybody tell me how to create coordinates to a drawing using scripts based on a given set of coordinates.

KlausDE at 2/9/2006 12:28 PM (#18913)
I assume you want to add Points to the ObjectSet of a Drawing.
How is input organised? Coming from a table or given by a function of a script? Coming as pairs numeric coordinates or as X/Y-columns or as single Geoms or as PointSet? Sample snippet

```
' VBscript
Set doc = Application.ActiveDocument
Set comps = doc.ComponentSet
Set drwng = comps("Drawing")
Set objcts = drwng.ObjectSet
' ...
x = 4
y = 6.54
Set Pnt = Application.NewPoint( x, y )
'
Set Geom = Application.NewGeom(GeomPoint, Pnt)
objcts.add( Geom )
```

- Klaus

Creating a New Geom and Adding it to a Drawing ObjectSet

http://69.17.46.171/Site/Thread.aspx?id=19973&ti=632792795307870000

aforest at 3/28/2006 10:10 PM (#19974)
How do I create a rectangle with give x and y min and max?
When I used the script below, I don't see the rectangle.

Set newrect = application.NewRect(newxmin, newymin, newxmax, newymax)
Can anyone tell me whats wrong in this script?

Thanks

KlausDE at 3/29/2006 2:24 AM (#19976)

What you see in a Drawing is a Set of Objects - not just any graphical representation stored in the application. Get the ObjectSet of the Drawing and add a Geom. To create the Geom you have to fill the PointSet with the coordinates of the Rectangle

```
-- VBScript

Set PSet = application.NewPointSet
PSet.add( application.newPoint( newxmin, newymin))
PSet.add( application.newPoint( newxmin, newymax))
...
PSet.add( application.newPoint( newxmin, newymin))
Set NewG = application.NewGeom(PSet)

Set comps = application.components
Set drwng = comps("Name of Drawing")
Set OSet = drwng.ObjectSet
OSet.Add( NewG )
```

(not tested) -- Klaus

mdsumner at 3/29/2006 4:18 PM (#20005)

A tested version (with an example) - you need an existing Drawing:

```
Sub Main

newxmin = -10
newxmax = 10
newymin = -10
newymax = 10
Set PSet = application.NewPointSet
PSet.add( application.newPoint( newxmin, newymin))
PSet.add( application.newPoint( newxmax, newymin))
PSet.add( application.newPoint( newxmax, newymax))
PSet.add( application.newPoint( newxmin, newymax))
Set NewG = application.NewGeom(GeomArea, PSet)

Set comps = application.ActiveDocument.componentSet
Set drwng = comps("Drawing")
Set OSet = drwng.ObjectSet
OSet.Add( NewG )

End Sub
```

;) There may well be a better way to achieve what you want if you give us more context.

aforest at 3/29/2006 5:14 PM (#20007)

Thanks a lot for your help. Sorry for not being more clearer. Since you have given me a sample of the script but i still want to know how do u used this syntext

Rect NewRect(Number mixX, Number minY, Number maxX, Number maxY) which i got it from the help manual. It just says creates new rectangle object with specified coordinates. So when i used this i dont c the graphical representation of the rectangle. If i m not mistaken it just returns the bounding box. So what additional script do i need to add in order to see the rectangle? Thanks

mdsumner at 3/29/2006 7:01 PM (#20009)

You have to add the bounding box as implicit in the Rect object as a Geom object to a drawing, then view the drawing.

If you create the Rect first, you would then obtain the bounding coordinates by querying its Xmin, Xmax, etc. properties.

```
newrect = application.NewRect(newxmin, newymin, newxmax, newymax)
Set PSet = application.NewPointSet
PSet.add( application.newPoint( newrect.xmin, newrect.ymin))
PSet.add( application.newPoint( newrect.xmax, newrect.ymin))
```

... etc. and proceed as in the script above.

Rect is really for obtaining bounding boxes, which are then use for querying existing objects or for specifying viewports in maps etc. and passing to image rendering methods to set the scope.

Extract Object Coordinates via Script Driven SQL

Code below is Not Form driven. For both Forms and non-form versions see:
C:\Documents and Settings\Lorne_Limited\My Documents\Manifold\VBscript\Extract_Object_Coords.map

```
' ********************************************************************

Option Explicit

' ********************************************************************

Sub Main()
  Dim Comps, Drwg

      ' Author: Lorne Ketch
      ' Permission to use, modify and distribute this code is granted,
      ' provided that you cite the original author.
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

  Set Comps = Document.ComponentSet
  Set Drwg = Comps("Drawing 2")

  Build_Run_SQL Drwg, 8

End Sub

' ********************************************************************
```

```
Private Sub Build_Run_SQL (ByRef Drwg, ByRef Mode)

      ' Author: Lorne Ketch
      ' Permission to use, modify and distribute this code is granted,
      ' provided that you cite the original author.
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

   Dim Comps, Qry
   Dim SQL, Where_Clause, Rcrds

    ' Mode: Which Objects to process.
    ' 1 = All
    ' 2 = Selection
    ' 3 = Points
    ' 4 = Lines
    ' 5 = Areas
    ' 6 = Points and Lines
    ' 7 = Points and Areas
    ' 8 = Lines and Areas

   Set Comps = Document.ComponentSet

   If Component_Exists("Query") Then
      Set Qry = Comps("Query")
   Else
      Set Qry = Document.NewQuery("Query")
   End If

   If Mode = 1 Then
     Drwg.SelectNone
     Where_Clause = ""

   ElseIf Mode = 2 Then
     Qry.Text = "SELECT * FROM [" & Drwg.Name & "] WHERE [Selection (I)];"
     Qry.RunEX TRUE
     Set Rcrds = Qry.Table.RecordSet
     If Rcrds.Count < 1 Then
       Application.MessageboxEX _
         "Manually select one or more objects from drawing " & VbCrLf & _
         """" & Drwg.Name & """ before running this option.", _
         "Not Ready", _
         MessageBoxTypeOK OR MessageBoxTypeSystemModal
       Drwg.Open
       Exit Sub
     End If

     Where_Clause = "WHERE [Selection (I)]"

   ElseIf Mode > 2 Then

     Select Case Mode
        Case 3
           Where_Clause = "Where [Type (I)] = 1"
        Case 4
           Where_Clause = "Where [Type (I)] = 2"
        Case 5
```

```
            Where_Clause = "Where [Type (I)] = 3"
         Case 6
            Where_Clause = "Where [Type (I)] = 1 OR [Type (I)] = 2"
         Case 7
            Where_Clause = "Where [Type (I)] = 1 OR [Type (I)] = 3"
         Case 8
            Where_Clause = "Where [Type (I)] = 2 OR [Type (I)] = 3"
      End Select

   End If

  Qry.Text = _
     "OPTIONS CoordSys(""" & Drwg.Name & """ AS COMPONENT);" & VbCrLf & _
     "SELECT ID, Cast([Type (I)] as Text) As Object_Type, " & VbCrLf & _
     "CentroidX([P]) as X, CentroidY([P]) as Y " & VbCrLf & _
     "FROM [" & Drwg.Name & "]" & VbCrLf & _
      Where_Clause & " SPLIT BY Coords(ID) AS [P];"

  Qry.RunEX TRUE

  Set Rcrds = Qry.Table.RecordSet
  If Rcrds.Count < 1 Then
     Application.Messagebox Drwg.Name & ": No Objects were Selected"
     Exit Sub
  End If

  Qry.Table.Open
  Qry.Open
  Drwg.SelectNone

End Sub

' ************************************************************************

Private Function Component_Exists(ByVal ComponentName)

  Dim Component_Index
  Dim Comps

      ' Author: Lorne Ketch
      ' Permission to use, modify and distribute this code is granted,
      ' provided that you cite the original author.
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

  Set Comps = Document.ComponentSet

  Component_Index = Comps.ItemByName(ComponentName)
  If Component_Index < 0 Then
    Component_Exists = False
  Else
    Component_Exists = True
  End If

End Function

' ************************************************************************
```

Extract Object Coordinates via Object Model

Code below is Not Form driven. For both Forms and non-form versions see:
C:\Documents and Settings\Lorne_Limited\My Documents\Manifold\VBscript\Extract_Object_Coords.map

```
' ********************************************************************************

Option Explicit

Sub Main()
  Dim Comps, Drwg

  Set Comps = Document.ComponentSet
  Set Drwg = Comps("Drawing 2")

  WriteCoords_2Comment Drwg, 8, 5

End Sub

' ********************************************************************************

Private Sub WriteCoords_2Comment (ByRef Drwg, ByRef Mode, ByRef NumberOfDecimals)

      ' Author: Lorne Ketch
      ' Permission to use, modify and distribute this code is granted,
      ' provided that you cite the original author.
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

  Dim Comps
  Dim Obj, Objs, Obj_Index
  Dim Obj_Geom, Obj_Branch, Branch_Index, Point_Index
  Dim Pt, Pts
  Dim Object_Type, Object_ID
  Dim Comment_1, Comment_2
  Dim Coord_Length, Formatted_Str
  Dim CoordSys, CoordSys_ParmSet
  Dim Last_Index
  Dim Obj_Count, Obj_Type
  Dim Qry, Rcrds
  Dim Selected_Objects
  Dim Where_Clause, Header
  Dim MaxX_WholeNumber, MaxY_WholeNumber

    '   Mode: Which Objects to process.
    ' 1 = All
    ' 2 = Selection
    ' 3 = Points
    ' 4 = Lines
    ' 5 = Areas
    ' 6 = Points and Lines
    ' 7 = Points and Areas
    ' 8 = Lines and Areas
    '
    ' NumberOfDecimals: Round coordinates to these many decimals
```

```
Set Comps = Document.ComponentSet
Set CoordSys = Drwg.CoordinateSystem
Set CoordSys_ParmSet = CoordSys.ParameterSet

If Component_Exists("Query") Then
   Set Qry = Comps("Query")
Else
   Set Qry = Document.NewQuery("Query")
End If

  ' Comment_1 will hold a formatted coordinated list
If Component_Exists("Object Coords 1") Then
   Set Comment_1 = Comps("Object Coords 1")
Else
   Set Comment_1 = Document.NewComments("Object Coords 1")
End If

  ' Comment_2 will hold an unformatted coordinated list.
  ' Just X and Y's - tab delimited - will past into Excel
  ' and will be automatically parsed to separate cells.
If Component_Exists("Object Coords 2") Then
   Set Comment_2 = Comps("Object Coords 2")
Else
   Set Comment_2 = Document.NewComments("Object Coords 2")
End If

If Mode = 1 Then
  Set Objs = Drwg.ObjectSet
  Selected_Objects = "Objects"

  Qry.Text = "Update (Select [Selection (I)] as Selection " & _
             "From [" & Drwg.Name & "]) " & _
             "Set Selection = TRUE;"
  Qry.RunEX TRUE

ElseIf Mode = 2 Then
  Qry.Text = "SELECT * FROM [" & Drwg.Name & "] WHERE [Selection (I)];"
  Qry.RunEX TRUE
  Set Rcrds = Qry.Table.RecordSet
  If Rcrds.Count < 1 Then
    Application.MessageboxEX _
      "Manually select one or more objects from drawing " & VbCrLf & _
      """" & Drwg.Name & """ before running this option.", _
      "Not Ready", _
      MessageBoxTypeOK OR MessageBoxTypeSystemModal
    Drwg.Open
    Exit Sub
  End If

  Set Objs = Drwg.Selection
  Selected_Objects = "Selected Objects"

ElseIf Mode > 2 Then
  Drwg.SelectNone
  Where_Clause = ""
```

```
      Select Case Mode
         Case 3
            Where_Clause = "Where [Type (I)] = 1"
            Selected_Objects = "Point Objects"
         Case 4
            Where_Clause = "Where [Type (I)] = 2"
            Selected_Objects = "Line Objects"
         Case 5
            Where_Clause = "Where [Type (I)] = 3"
            Selected_Objects = "Area Objects"
         Case 6
            Where_Clause = "Where [Type (I)] = 1 OR [Type (I)] = 2"
            Selected_Objects = "Point and Line Objects"
         Case 7
            Where_Clause = "Where [Type (I)] = 1 OR [Type (I)] = 3"
            Selected_Objects = "Point and Area Objects"
         Case 8
            Where_Clause = "Where [Type (I)] = 2 OR [Type (I)] = 3"
            Selected_Objects = "Line and Area Objects"
      End Select

      Qry.Text = "Update (Select [Selection (I)] as Selection " & _
                 "From [" & Drwg.Name & "] " & Where_Clause & " ) " & _
                 "Set Selection = TRUE;"
      Qry.RunEX TRUE

    End If

   Set Objs = Drwg.Selection

   Get_Coord_WholeNumber_MaxStrLength Drwg, MaxX_WholeNumber, MaxY_WholeNumber

   If uCase(Selected_Objects) = "OBJECTS" Then
      Header = "USER REQUESTED ALL " & uCase(Selected_Objects)
   Else
      Header = "USER REQUESTED " & uCase(Selected_Objects)
   End IF

   If Objs.Count > 0 Then
      Comment_1.Clear

      Comment_1.AddText Header & VbCrLf & VbCrLf

      Comment_1.AddText "Drawing name:     " & Drwg.Name & VbCrLf
      Comment_1.AddText "CoordSys:         " & CoordSys.Name & VbCrLf
      Comment_1.AddText "Datum:            " & CoordSys.Datum.Name & VbCrLf
      Comment_1.AddText "EllipsoidName:    " & CoordSys.Datum.Ellipsoid.Name & VbCrLf

      Comment_1.AddText "CoordSys Units:   " & CoordSys.Unit.Name & VbCrLf

      Comment_1.AddText "localOffsetX:     " &
Cstr(CoordSys_ParmSet.Item("localOffsetX").Value) & VbCrLf
      Comment_1.AddText "localOffsetY:     " &
Cstr(CoordSys_ParmSet.Item("localOffsetY").Value) & VbCrLf

      Comment_1.AddText "localScaleX:      " &
Cstr(CoordSys_ParmSet.Item("localScaleX").Value) & VbCrLf
```

```
     Comment_1.AddText "localScaleY:      " &
Cstr(CoordSys_ParmSet.Item("localScaleY").Value) & VbCrLf

     Comment_1.AddText "ScaleCorrectionX: " &
Cstr(CoordSys_ParmSet.Item("ScaleX").Value) & VbCrLf
     Comment_1.AddText "ScaleCorrectionY: " &
Cstr(CoordSys_ParmSet.Item("ScaleY").Value) & VbCrLf

     Comment_1.AddText "FalseEasting:     " &
Cstr(CoordSys_ParmSet.Item("FalseEasting").Value) & VbCrLf
     Comment_1.AddText "FalseNorthing:    " &
Cstr(CoordSys_ParmSet.Item("FalseNorthing").Value) & VbCrLf

     Comment_1.AddText "majorAxis:        " &
Cstr(CoordSys_ParmSet.Item("majorAxis").Value) & VbCrLf
     Comment_1.AddText "Eccentricity:     " &
Cstr(CoordSys_ParmSet.Item("Eccentricity").Value) & VbCrLf & VbCrLf

     Comment_1.AddText "Number of " & Selected_Objects & " = " & Objs.Count & VbCrLf &
VbCrLf

     Comment_2.Clear
     Comment_2.AddText Header & VbCrLf & VbCrLf
  Else
     Application.Messagebox """" & DrawingName & _
         """ does not contain any " & Selected_Objects & "."
     Form.Visible = TRUE
     Exit Sub
  End If

  Last_Index = Objs.Count - 1
    ' Examine each object.
  For Obj_Index = 0 To Last_Index

    Set Obj = Objs.Item(Obj_Index)
    Set Obj_Geom = Obj.Geom

    Object_Type = Obj.TypeName
    Object_ID = cStr(Objs.OwnerComponent.OwnedTable.RecordSet(Obj_Index).Data("ID"))
    Comment_1.AddText "Object Type........." & Object_Type & VbCrLf
    Comment_1.AddText "Object ID............." & Object_ID & VbCrLf
    Comment_1.AddText "Number of Branches...." & Obj_Geom.BranchSet.Count & VbCrLf

    For Branch_Index = 0 to Obj_Geom.BranchSet.Count - 1
      Comment_1.AddText "Branch " & Branch_Index & ":" & VbCrLf

           ' Look at each Branch in the Branch Set
      Set Obj_Branch = Obj_Geom.BranchSet.Item(Branch_Index)

            ' Get the point set for this particular branch
      Set Pts = Obj_Branch.PointSet

            ' Get the point coordinates for this Point Set
      For Point_Index = 0 To Pts.Count - 1

         Set Pt = Pts.Item(Point_Index)
         Formatted_Str =  Format_Coordinates (Pt.X, Pt.Y, _
```

```
                MaxX_WholeNumber, MaxY_WholeNumber, NumberOfDecimals)
          Comment_1.AddText Formatted_Str & VbCrLf
          Comment_2.AddText FormatNumber(Pt.X, NumberOfDecimals, -1) & VbTab & _
                            FormatNumber(Pt.Y, NumberOfDecimals, -1) & VbCrLf

      Next   ' Next Point_Index

    Next     ' Next Branch_Index

    Comment_1.AddText VbCrLf

    Application.StatusText = cstr(Obj_Index + 1) & " of " & cStr(Last_Index + 1)

  Next       ' Next Obj_Index

  Application.StatusText = ""
  Comps("Object Coords 2").Open
  Comps("Object Coords 1").Open

  Drwg.SelectNone

End Sub

' ************************************************************************

Private Function Component_Exists(ByVal ComponentName)

  Dim Component_Index
  Dim Comps

      ' Author: Lorne Ketch
      ' Permission to use, modify and distribute this code is granted,
      ' provided that you cite the original author.
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

  Set Comps = Document.ComponentSet

  Component_Index = Comps.ItemByName(ComponentName)
  If Component_Index < 0 Then
    Component_Exists = False
  Else
    Component_Exists = True
  End If

End Function

' ************************************************************************

Private Function Format_Coordinates ( _
     ByVal X, _
     ByVal Y, _
     ByRef MaxX_WholeNumber, _
     ByRef MaxY_WholeNumber, _
     ByRef NumberOfDecimals)

      ' Author: Lorne Ketch
      ' Permission to use, modify and distribute this code is granted,
```

```
      ' provided that you cite the original author.
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

  Dim X_Str, Y_Str
  Dim Decimal_Position
  Dim MaxX_Str_Length, MaxY_Str_Length
  Dim Str_Length
  Dim Padding

  MaxX_Str_Length = cLng(MaxX_WholeNumber + NumberOfDecimals + 1)
  MaxY_Str_Length = cLng(MaxY_WholeNumber + NumberOfDecimals + 1)

     ' DO THE X
  X = FormatNumber(X, NumberOfDecimals, -1)

  Str_Length = Len(X)

  Padding = ""
  If Str_Length < MaxX_Str_Length Then
    Padding = String(MaxX_Str_Length - Str_Length, 32)
  End If

  X_Str = "X: " & Padding & X


       ' DO THE Y
  Y = FormatNumber(Y, NumberOfDecimals, -1)

  Str_Length = Len(Y)

  Padding = ""
  If Str_Length < MaxY_Str_Length Then
    Padding = String(MaxY_Str_Length - Str_Length, 32)
  End If

  Y_Str = "Y: " & Padding & Y

  Format_Coordinates = X_Str & "  " & Y_Str

End Function

' ********************************************************************

Private Function Get_Coord_WholeNumber_MaxStrLength( _
        ByRef Drwg, _
        ByRef MaxX_WholeNumber, _
        ByRef MaxY_WholeNumber)

      ' Author: Lorne Ketch
      ' Permission to use, modify and distribute this code is granted,
      ' provided that you cite the original author.
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

     ' Comp is a Drawing Component.
     ' We want to find the coordinate in the Drawings Object set
     ' that contains the largest number of characters and
     ' return the string length of that coordinate.
```

```
      ' This would include any leading minus sign and
      ' a decimal point.

   Dim Rect
   Dim Str_Lengths(3)
   Dim Index

   Set Rect = Drwg.ObjectSet.GeomSet.Box

   Str_Lengths(0) = Len(cStr(Int(Rect.XMax)))
   Str_Lengths(1) = Len(cStr(Int(Rect.XMin)))
   Str_Lengths(2) = Len(cStr(Int(Rect.YMax)))
   Str_Lengths(3) = Len(cStr(Int(Rect.YMin)))

   If Str_Lengths(0) >= Str_Lengths(1) Then
      MaxX_WholeNumber = Str_Lengths(0)
   Else
      MaxX_WholeNumber = Str_Lengths(1)
   End If

   If Str_Lengths(2) >= Str_Lengths(3) Then
      MaxY_WholeNumber = Str_Lengths(2)
   Else
      MaxY_WholeNumber = Str_Lengths(3)
   End If

End Function

' ********************************************************************************
```

Tables

Add Columns to a Table

http://69.17.46.171/Site/Thread.aspx?id=17410&ti=632713376167300000

Rodrigo at 12/28/2005 12:15 AM (#17411)

I'm programming a script in Manifold 6, an I've achieved to create a NewDrawing by code selecting records from another drawing. But this new drawing has only its own field (ID), I want to generate this drawing with more columns. This is my code:

```
Sub Main

' -- create temporary query component
Set document = Application.ActiveDocument
Set components = document.ComponentSet

set columns = Application.NewColumnSet
set column = columns.NewColumn
column.Name = "col_1"
column.Type = ColumnTypeWText
columns.Add column


Set dibujo = document.NewDrawing("dibujo")

Set drawingA = components(components.ItemByName("MFE3"))
Set drawingB = components(components.ItemByName("Dibujo"))

Set objectsB = drawingB.ObjectSet

Set recordsA = drawingA.OwnedTable.RecordSet
Set recordsB = drawingB.OwnedTable.RecordSet


For index = 0 To drawingA.ObjectSet.Count-1
Set object = drawingA.ObjectSet(index)


Set recordA = recordsA(recordsA.ItemByID(object.ID))
Set recordB = recordsB(recordsB.ItemByID(objectsB.LastAdded.ID))

if ((recordA.Data("SP1") = 45) or (recordA.Data("SP2") = 45) or (recordA.Data("SP3") =
45)) then
objectsB.Add object.Geom
'recordB.Data("Name") = recordA.Data("Name")
end if

Next

'query.Text = "Select * FROM [MFE3] WHERE [SP1] = 21 OR [SP2] = 21 OR [SP3] = 21;"

'Set table = query.Table
```

```
End Sub
Rodrigo at 12/28/2005 3:33 AM (#17412)
Solution:

set cols = Application.NewColumnSet
set col = cols.NewColumn
col.Name = "IPM"
col.Type = ColumnTypeWText
'cols.Add col


Set dibujo = document.NewDrawing("dibujo")

Set tabladibujo = components.Item("Dibujo Table")
Set columnsdibujo = tabladibujo.ColumnSet
columnsdibujo.Add(col)
```

Adding Columns to a Table via Script / Manifold© COLUMN ENUMERATIONS

```
     ' obtain drawing table and add a 'Name' column
  Set DrawingTable = Drawing.OwnedTable
  Set NewColumn    = DrawingTable.ColumnSet.NewColumn
  NewColumn.Name   = "Name"
  NewColumn.Type   = ColumnTypeWText  ' (see enumerations below)
  DrawingTable.ColumnSet.Add(NewColumn)
```

To insert data into the new column, use the Data property:

```
'  Insert data for the last record in the recordset
Set Last_Record  = drawingTable.RecordSet.Count - 1)
  Or
    Set Last_Record = drawingTable.LastAdded
Last_Record.Data("Name") = Name

     '  *****  MANIFOLD© COLUMN ENUMERATIONS'  *****
     '  ColumnTypeAChar      - ANSI character.
     '  ColumnTypeAText      - ANSI text (variable-length or fixed-length).
     '  ColumnTypeBinary     - Binary data (variable-length or fixed-length).
     '  ColumnTypeBoolean    - Boolean value.
     '  ColumnTypeCoordSys   - Coordinate system.
     '  ColumnTypeCurrency   - Currency.
     '  ColumnTypeFloat32    - Single precision floating-point number.
     '  ColumnTypeFloat64    - Double precision floating-point number.
     '  ColumnTypeGeom       - Geometric shape.
     '  ColumnTypeGeomSHP    - Geometric shape in ESRI SHP format.
     '  ColumnTypeGeomWKB    - Geometric shape in OpenGIS WKB format.
     '  ColumnTypeInt16      - 16 bit integer number.
     '  ColumnTypeInt16U     - Unsigned 16 bit integer number.
     '  ColumnTypeInt32      - 32 bit integer number.
     '  ColumnTypeInt32U     - Unsigned 32 bit integer number.
     '  ColumnTypeInt8       - 8 bit integer number.
     '  ColumnTypeInt8U      - Unsigned 8 bit integer number.
     '  ColumnTypeLatitude   - Latitude value.
```

```
'  ColumnTypeLongitude  - Longitude value.
'  ColumnTypePercentage - Percentage value.
'  ColumnTypeTime       - Date and time value.
'  ColumnTypeURL        - URL string.
'  ColumnTypeWChar      - Unicode character.
'  ColumnTypeWText      - Unicode string.
```

Scripting Table Relationships

http://69.17.46.171/Site/Thread.aspx?id=28521&ti=632955606300630000

artlembo on 10/4/2006 7:41 AM (#28522)

I've been able to create a relation via a script. However, I don't know where one turns on the actual columns to view the columns in the related table.

RelationSet.Add allows you to create the relation, but I can't find out how to then tell Manifold to include the specific columns when viewing the related table.

willh on 10/4/2006 11:09 AM (#28536)

I was going to post an answer to this the other day when I saw your earlier post but I guess I misunderstood that you had sorted it out. In ScriptTableNwind.Map, the lines below add the "mapped" columns after the relation has been created. The key items are creating a new column object (under the ColumnSet of the table you're working on) and setting the OriginColumn property

```
' map products column
set columnProductsMap = columnsCategories.NewColumn
nItem = columnsProducts.ItemByName( "Product Name" )
columnProductsMap.Name = "Sample Product"
columnProductsMap.OriginColumn = columnsProducts.Item( nItem )
columnsCategories.Add(columnProductsMap)
```

artlembo on 10/4/2006 11:14 AM (#28537)

thanks Will. I was hoping there was something easier. I have like 7 different columns to add. And, sometimes I don't necessarily know what each of the column names are beforehand.

I put in a request to Manifold hoping there would be something like a "RelateAll", "RelateSome", "RelateNone" option.

I don't like the current way its being done.

thanks again.

willh on 10/4/2006 12:10 PM (#28539)

Some refactoring could give you a generic Sub like the one below. Then you could loop though your ColumnSet and use column properties like IsIntrinsic to filter out unwanted columns.

```
'to keep inline with the earlier example, it would be called like this

addRelCol columnsProducts, columnsCategories, "Product Name"

Sub addRelCol(sourceColSet, targetColSet, colName)
  set colMap = targetColSet.NewColumn
  colSrc = sourceColSet.ItemByName(colName)
  colMap.Name = colName
  colMap.OriginColumn = sourceColSet.Item(colSrc)
  targetColSet.Add(colMap)
End Sub
```

Access Column Data by Index Rather than Column Name

http://69.17.46.171/Site/Thread.aspx?id=18333&ti=632741989491870000

KlausDE at 1/27/2006 5:24 AM (#18337)

You can access data by column index - i.e. Record.Data(Number ID) - and you could sum data in a loop:

```
For ID = 4 to 17
    sum = sum + Record.Data(ID)
next id
```

This would speed up your coding at the cost of slowing down the calculation of the active column. – Klaus

WillH on 1/30/2006 6:20 AM (#18401)

That might still be a bit cumbersome if the drawing is worked on (transforms, additional objects, etc) ***in which case the IDs would regularly get changed*** ; why not base the loop on RecordSet.Count?

Will

adamw on 1/30/2006 6:22 AM (#18404)

The request is to sum a set of columns, not a set of records. :-)

adamw on 1/30/2006 6:21 AM (#18403)

Using IDs might not be the best choice since the user has no control over them.

If the column names use a specific pattern, you could try something like:

```
'VBScript
sum = 0
For i = 1 To N
      ' "Column1" for column number 1, etc
  sum = sum + Record.Data("Column" & CStr(i))
Next
```

If there is no pattern, try:

```
'VBScript
names = Array("A", "B", "C")
sum = 0
For i = LBound(names) to UBound(names)
  sum = sum + Record.Data(names(i))
Next
```

Finding and Deleting Duplicate Records based upon Column Contents

http://69.17.46.171/Site/Thread.aspx?id=20092&ti=632803720515300000

NOTE 1: ColinD posted the original question on GeoReference. Following is a summary of some of the responses.
NOTE 2: See Adam's SQL approach at the end.

Remove Duplicates, when used on a Drawing, removes Objects (rows) with identical geometries. It doesn't care about the columns in the drawing--it ignores them entirely and "simply removes the most recently-created objects".

KlausDE at 4/9/2006 1:55 AM (#20326)

You must get rid of the unique ID for the DISTINCTROW. i.e you cannot use * nor select objects in the original Drawing. The following script creates a query with all non-intrinsic columns and excludes duplicates. Delete the columns you don't need and link a Drawing to the Query. Unlink and you're done.

```
-- VBScript
Sub Main
' KlausDE at 4/9/2006 2:02 AM (#20327)
' Let me add a warning: I'm not sure if this query will use Precision to
' check for +/- identical Geoms. I.e. if you have created duplicates by
' means involving any mathematical process like projecting the objects
' will probably appear different
'
' adamw at 4/10/2006 7:07 AM (#20351)
' It won't. The comparison of the geoms will be exact.

DrwngTNm = "Drawing Table"

SET doc = Application.ActiveDocument
SET comps = doc.ComponentSet
SET DrwngTbl = comps(DrwngTNm)
SET Clmns = DrwngTbl.ColumnSet

   ' Start building the SQL string
Text = "SELECT DISTINCTROW Geom([ID]) AS Gm"

FOR EACH Clmn IN Clmns
```

```
     ' Add the columns to the SQL string
    ' Line below was Klaus' original
'  If NOT((Right(Clmn.Name,3)="(I)") OR Clmn.Name = "ID") then'
   ' Line below was suggested by Adam as a more robust alternative
  If Not clmn.IsIntrinsic() And Not clmn.Identity Then
   Text = Text + ",[" + Clmn.Name + "]"
  end if
NEXT

   ' Finish the SQL string
Text = Text + " FROM [" + DrwngTNm + "]"

Set qr = doc.NewQuery("SelectUnique "+DrwngTNm, FALSE)
qr.Text = Text

Application.MessageBox "done.", "Script"

End Sub
```

-- Klaus

adamw at 4/10/2006 7:07 AM (#20352)

You might want to modify the condition in the line filtering the intrinsic columns and the ID column to:

```
'VBScript
If Not clmn.IsIntrinsic() And Not clmn.Identity Then
```

This will make the code more robust.

Under certain circumstances, the name of the ID column might differ from "ID" and the names of the intrinsic columns might not end with "(I)".

adamw at 4/10/2006 7:07 AM (#20353)

An alternative idea is to create a query to select the duplicate objects, eg:

```
--SQL
SELECT * FROM [Drawing] WHERE [ID] NOT IN
  (SELECT First([ID]) FROM [Drawing] GROUP BY [Column1], [Column2], ...)
```

Save Drawing Names to a Column

ColinD on 12/11/2006 2:13 AM (#31824)

I have around 700 drawings (imported by script) in a folder in a map project and want to get the drawing name into a field in a column called Name for each. Could someone please point me in the right direction to do this. To complicate things I just need the 'name' without the appended 'drawing' that comes with importing a drawing. Thanks.

Lorne on 12/11/2006 4:35 AM (#31829)

See the attachment: DrawingNames_2_TableColumn.map

Option Explicit

```
'*************************************************************

Sub Main
   Dim Comp, Comps
   Dim Cols, Col
   Dim Qry
   Dim Tbl, Rcrd, Rcrds

     ' Lorne Ketch - Dec 11, 2006
     ' LKetch|at|hfx[1st_dot]eastlink{2nd_dot}ca
     '
     ' Collects Drawing names and saves them to:
     ' Table: "Drawing Names"  Column: "Name"
     '
     ' NOTE: If the Drawing Name ends with
     '       " Drawing", that portion of the
     '       name is deleted.
     '       Example:
     '       "My Drawing" becomes "My"
     '       "My Drawing 2" will not be changed.

   Set Comps = Document.ComponentSet

      ' "Drawing Names" = One-column table
      '                   Column Name = "Table"
      '                   Column Type = ColumnTypeAText
   If Comps.ItemByName("Drawing Names") < 0 Then
         ' Create a new Table
      Set Cols = Application.NewColumnSet
      Set Col = Cols.NewColumn
      Col.Name = "Name"
      Col.Type = ColumnTypeWText
      Cols.Add(Col)
      Set Tbl = Document.NewTable("Drawing Names", Cols, TRUE)
      Set Rcrds = Tbl.RecordSet
   Else
      Set Tbl = Comps.Item("Drawing Names")
         ' Delete any existing records.
      Set Rcrds = Tbl.RecordSet
      Rcrds.RemoveAll
   End If

    ' Collect Drawing component names (minus the " Drawing")
  For Each Comp in Comps
    If Comp.Type = ComponentDrawing Then
       Rcrds.AddNew
       If Len(Comp.Name) > 8 And uCase(Right(Comp.Name, 8)) = " DRAWING" Then
          Rcrds.LastAdded.Data("Name") = Left ( Comp.Name, Len(Comp.Name) - 8 )
       Else
          Rcrds.LastAdded.Data("Name") = Comp.Name
```

```
        End If
     End If
  Next

End Sub

' ************************************************************
```

ColinD on 12/11/2006 1:06 PM (#31862)

Thanks Lorne.

It's only when someone else reads a post that you realise that you as the original poster knew, but didn't properly say, what you meant!

What I meant to say was that I needed each drawing name to be put into a new column in that same drawing. The reason being that I am creating a single drawing of common bounding boxes for each individual drawing and want the name to automatically come into the boxes table- creating a tile index for the drawings.

```
Option Explicit

' ************************************************************

Sub Main
   Dim Comps, Comp, Tbl
   Dim Cols, Col
   Dim Col_Name, Drwg_Name
   Dim Qry

     ' Lorne Ketch - Dec 11, 2006
     ' LKetch|at|hfx[1st_dot]eastlink{2nd_dot}ca
     '
     ' For each Drawing Component in the Project:
     '   Inserts a column named "Name"
     '   Fills that column with the Drawing Name
     '
     ' NOTE: If the Drawing Name ends with
     '       " Drawing", that portion of the
     '       name is deleted.
     '       Example:
     '        "My Drawing" becomes "My"
     '        "My Drawing 2" will not be changed.

   Set Comps = Document.ComponentSet

      ' Make sure the Query component exists
   If Comps.ItemByName("Query") < 0 Then
      Set Qry = Document.NewQuery("Query")
   Else
      Set Qry = Comps.Item("Query")
   End If
```

```
   For Each Comp in Comps
     If Comp.Type = ComponentDrawing Then
        Set Tbl   = Comp.OwnedTable
        Set Cols  = Tbl.ColumnSet
        Set Col   = Cols.NewColumn
        Col.Name  = "Name"
        Col.Type  = ColumnTypeWText
        Cols.Add(Col)

           ' Collect the column name just in case column "Name"
           ' already exists in the table and Manifold automatically
           ' changes the name.
        Col_Name = Cols.LastAdded.Name

        If Len(Comp.Name) > 8 And uCase(Right(Comp.Name, 8)) = " DRAWING" Then
           Drwg_Name = Left ( Comp.Name, Len(Comp.Name) - 8 )
        Else
           Drwg_Name = Comp.Name
        End If

        Qry.Text = "Update [" & Tbl.Name & "] Set [" & Col_Name & "] = """ & Drwg_Name
& """;"
        Qry.RunEX TRUE

     End If
   Next

End Sub

' ************************************************************
```

Return the Identity Column Name

```
Private Function Identity_Column_Name(ByRef Drwg)
      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

      ' The ID (Identity) column many not always be labelled as "ID".
      ' The function locates the correct Identity column.
      ' Adam explains this in the GeoReference thread below.
      '
      ' adamw at 4/10/2006 7:07 AM (#20352)
      ' http://69.17.46.171/Site/Thread.aspx?id=20092&ti=632803720515300000
      ' You might want to modify the condition in the line filtering the
      ' intrinsic columns and the ID column to:
      '  VBScript
      ' If Not clmn.IsIntrinsic() And Not clmn.Identity Then

      ' This will make the code more robust.
      ' Under certain circumstances, the name of the ID column might differ
      ' from "ID" and the names of the intrinsic columns might not end with "(I)".

   Dim Col, Cols
   Set Cols =  Drwg.OwnedTable.ColumnSet
```

```
   Identity_Column_Name = ""
   For Each Col in Cols
      If Col.Identity Then
         Identity_Column_Name = Col.Name
         Exit Function
      End If
   Next

End Function

' ************************************************************************
```

Collect Collumn Names (All, Regular, Intrinsics only, etc)

C:\Documents and Settings\Lorne_Limited\My Documents\Manifold\VBscript\Retrieve_Column_Names.map

```
Option Explicit

' ************************************************************************

Sub Main
   Dim Comps, Tbl
   Dim Column_Names()
   Dim Index
   Dim Report
   Dim OK

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

   Set Comps = Document.ComponentSet
   Set Tbl = Comps("HURDAT_2000to2005 Table")

   OK = Collect_Column_Names( 3, Tbl, Column_Names, FALSE, FALSE )
   If NOT OK Then Exit Sub

   For Index = 0 to uBound(Column_Names)
     Report = Report & Column_Names(Index) & VbCrLf
   Next

   Report = Report &  VbCrLf & "Identity Column:" & VbCrLf & _
       Identity_Column_Name(Tbl)

   Application.Messagebox Report

End Sub

' ************************************************************************

Private Function Collect_Column_Names ( _
       ByRef Column_Class, _
       ByRef Tbl, _
       ByRef Column_List(), _
       ByRef Sort, _
       ByRef Descending)
```

```
      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca

      ' Column_Class VALUES:
      '   1. Regular Columns Only
      '   2. ID + Regular Columns
      '   3. All: ID + Regular Columns + Intrinsics
      '   4. Intrinsics Only
      '   5. ID Only (not the best function to extract a single value)
      ' Tbl:           a Manifold Table Object
      ' Column_List:   an unitialized Dynamic Array
      ' Sort:          sort the List using BubbleSort (TRUE or FALSE)
      ' Descending:    sort Descending (TRUE) or Ascending (FALSE)
      '
      ' NOTE:
      ' The ID (Identity) column many not always be labelled as "ID".
      ' and Intrinsics might not end in (I).
      ' Adam explains this in the GeoReference thread below.
      '
      ' adamw at 4/10/2006 7:07 AM (#20352)
      ' http://69.17.46.171/Site/Thread.aspx?id=20092&ti=632803720515300000
      ' You might want to modify the condition in the line filtering the
      ' intrinsic columns and the ID column to:
      '     If Not clmn.IsIntrinsic() And Not clmn.Identity Then
      ' This will make the code more robust.
      ' Under certain circumstances, the name of the ID column might
      ' differ from "ID" and the names of the intrinsic columns might
      ' not end with "(I)".

   Dim Index, Last_Index, Last_Array_Index
   Dim Cols, Column_Index, Mode_Error

   Collect_Column_Names = TRUE
   Mode_Error = FALSE

      ' If Column_Class is not numeric and we attempt to do
      ' an operation like Column_Class < 1, an error will
      ' be generated...
   If NOT IsNumeric(Column_Class) Then
       Mode_Error = TRUE
   ElseIf Column_Class < 1 Or Column_Class > 5 Then
       Mode_Error = TRUE
   End If

   If Mode_Error Then
       Application.MessageboxEX _
           "The first parametre passed to function ""Collect_Column_Names"" must " &
VbCrLf & _
           "be a number between 1 and 5.  That number specifies which column " &
VbCrLf & _
           "names to collect. You did not pass a number or the number is " & VbCrLf &
_
           "not in the correct range." & VbCrLf & VbCrLf & _
           "  1.  Regular Columns Only" & VbCrLf & _
           "  2.  ID + Regular Columns" & VbCrLf & _
           "  3.  ALL: ID + Regular Columns + Intrinsics" & VbCrLf & _
```

```
        "  4.  Intrinsics Only" & VbCrLf & _
        "  5.  ID Only",  _
        "Choose Columns Error", _
        MessageBoxTypeOK OR MessageBoxTypeIconError OR MessageBoxTypeSystemModal

    Collect_Column_Names = FALSE
    Exit Function
End If

If Tbl.Type <> ComponentTable Then
    Application.MessageboxEX _
        "You must supply a Table Component Object to function:" & VbCrLf & _
        "Collect_Column_Names", _
        "Not a Table", _
        MessageBoxTypeOK OR MessageBoxTypeIconError OR MessageBoxTypeSystemModal

    Collect_Column_Names = FALSE
    Exit Function
End If

Set Cols = Tbl.ColumnSet

ReDim Column_List (100)
Last_Array_Index = 100
Last_Index = Cols.Count - 1

Column_Index = -1

For Index = 0 to Last_Index

  If Index > Last_Array_Index Then
      ReDim Preserve Column_List( Last_Array_Index + 100)
      Last_Array_Index = uBound(Column_List)
  End If

  Select Case Column_Class
     Case 1 ' Regular Columns Only
        If NOT Cols(Index).IsIntrinsic() and NOT Cols(Index).Identity Then
            Column_Index = Column_Index + 1
            Column_List(Column_Index) = Cols(Index).Name
        End If
     Case 2 ' ID + Regular Columns
        If NOT Cols(Index).IsIntrinsic() Then
            Column_Index = Column_Index + 1
            Column_List(Column_Index) = Cols(Index).Name
        End If
     Case 3 ' ID + Regular Columns + Intrinsics (ALL)
        Column_Index = Column_Index + 1
        Column_List(Column_Index) = Cols(Index).Name
     Case 4 ' Intrinsics Only
        If Cols(Index).IsIntrinsic() Then
            Column_Index = Column_Index + 1
            Column_List(Column_Index) = Cols(Index).Name
        End If
     Case 5 ' ID Only
           ' Provided for completeness only.  It would be
           ' better to have a stand-alone function to
```

```
                ' return this single value since we must pass
                ' a dynamic array to this function.
                ' (No Optional parametres in VBscript)
                ' See Function "Identity_Column_Name"

            If Cols(Index).Identity Then
                Column_Index = Column_Index + 1
                Column_List(Column_Index) = Cols(Index).Name
                  ' We can get out since we only require one Column name.
                ReDim Preserve Column_List(Column_Index)
                Exit Function
            End If
      End Select

   Next

   ReDim Preserve Column_List(Column_Index)

   If Sort Then
      If Descending Then
        BubbleSort_uCaseText Column_List, TRUE
      Else
        BubbleSort_uCaseText Column_List, FALSE
      End If
   End If

End Function

' ****************************************************************************
```

Trim all Text Columns in a Table

A simpler form of this routine was posted at:
http://forum.manifold.net/Site/Thread.aspx?id=28218&ti=632967693061100000

```
Option Explicit

' ****************************************************************

Sub Main()
   Dim Comps, Qry
   Dim Tbl, Col, Cols

      ' Author: Lorne Ketch
      ' Email:  LKetch[at]hfx{dot}eastlink(.)ca


   Set Comps = Document.ComponentSet

      ' Make sure the Query component exists
  If Comps.ItemByName("Query") < 0 Then
     Set Qry = Document.NewQuery("Query")
  Else
```

```
     Set Qry = Comps.Item("Query")
  End If

   Set Tbl = Comps("My Table")
   Set Cols = Tbl.ColumnSet

   For Each Col in Cols
      If Not Col.IsIntrinsic() And _
         Not Col.Identity And _
            (Col.Type = ColumnTypeAText OR Col.Type = ColumnTypeWText) Then
         Qry.Text = "Update [" & Tbl.Name & "] Set [" & Col.Name & "] = TRIM([" &
Col.Name & "]);"
         Qry.RunEX TRUE
      End If
   Next

End Sub

' ************************************************************************
```

Create a Drawing from a Table that Contains a Manifold Geometry Column

http://forum.manifold.net/Site/Thread.aspx?id=31021

NOTE: The code below was not posted. This version automatically extracts the coordinate information from the first record of the table and uses it to build the new drawing

See: C:\Documents and Settings\Lorne_Limited\My Documents\Manifold\VBscript\CreateDrawing_FromTable.map

```
' ************************************************************************

Private Sub Convert_GeomTable2Drawing ( _
       ByRef SourceGeom_TableName, _
       ByRef Geom_ColumnName, _
       ByRef NewDrawing_Name)

      ' Author: Lorne Ketch
      ' Email:  LKetch|at|hfx[1st_dot]eastlink{2nd_dot}ca

      ' V1.0: Nov 29, 2006
      ' V1.1: Jan 12, 2007 - Added Functions:
      '                      Extract_CoordSys_From_Geom()
      '                      Table_Column_Exists()

   Dim Comps, Qry
   Dim Source_Tbl, Source_Col, Source_Cols
   Dim New_Drawing, New_Table, New_ColumnSet, New_Column
   Dim Column_List
   Dim Index, Last_Index
   Dim Coord_Sys

   Set Comps = Document.ComponentSet
```

```
    ' Make sure the Query component exists
   If Comps.ItemByName("Query") < 0 Then
       Set Qry = Document.NewQuery("Query")
   Else
       Set Qry = Comps.Item("Query")
   End If

   If NOT Table_Column_Exists(SourceGeom_TableName, Geom_ColumnName, ColumnTypeGeom)
Then Exit Sub

   Set Source_Tbl  = Comps.Item(SourceGeom_TableName)
   Set Source_Cols = Source_Tbl.ColumnSet
   Set Source_Col  = Source_Cols(Geom_ColumnName)

      ' Extract the Coordinate system from the Geom column of Record #1
   Set Coord_Sys = Extract_CoordSys_From_Geom(SourceGeom_TableName, Geom_ColumnName)

   Set New_Drawing   = Document.NewDrawing(NewDrawing_Name, Coord_Sys, FALSE)
   Set New_Table     = New_Drawing.OwnedTable
   Set New_ColumnSet = New_Table.ColumnSet

   Last_Index   = Source_Cols.Count - 1
   Column_List  = ""

   On Error Resume Next
   For Index = 0 to Last_Index
      Set Source_Col = Source_Cols(Index)
      If NOT Source_Col.IsIntrinsic() and _
         NOT Source_Col.Identity and _
         NOT Source_Col.Type = ColumnTypeGeom Then

         Set New_Column = New_ColumnSet.NewColumn
            ' Intrinsics and ID should not be in the table but just in case....
         New_Column.Name = Source_Col.Name
         New_Column.Type = Source_Col.Type
         New_Column.Size = Source_Col.Size
         New_ColumnSet.Add(New_Column)

              ' Assemble a Column List for the INSERT INTO Query
         If Index = Last_Index Then
            Column_List = Column_List & "[" &  Source_Col.Name & "]"
         Else
            Column_List = Column_List & "[" &  Source_Col.Name & "],"
         End If

     End If

   Next

   Set Qry = Comps.Item("Query")

   Qry.Text = "INSERT INTO [" & New_Table.Name & "] ([Geom (I)],"  & Column_List & _
              ") SELECT [" & Geom_ColumnName & "]," & Column_List & _
              " FROM [" & SourceGeom_TableName & "];"

   Qry.RunEX TRUE
```

```
   New_Drawing.CoordinateSystemVerified = TRUE

End Sub

' ********************************************************************

Private Function Extract_CoordSys_From_Geom( _
       ByRef SourceGeom_TableName, _
       ByRef Geom_ColumnName)

      ' Author: Lorne Ketch
      ' Email:  LKetch|at|hfx[1st_dot]eastlink{2nd_dot}ca

      ' Extracts the CoordSys from the First Record in the source geometry table.
      '
      ' NOTE: http://forum.manifold.net/Site/Thread.aspx?id=25835
      ' adamw on 8/14/2006 6:19 AM (#25878)
      ' Writing to the [Geom (I)] field of a particular drawing will accept values
      ' in any projection, but reading from the field will always return values in
      ' the projection of the drawing. You can have a table with a Geom field, or
      ' a query that returns a Geom field, and have EACH VALUE IN THAT FIELD BE
      ' IN A DIFFERENT PROJECTION THOUGH.

   Dim Comps, Qry, Tbl, Rcrds

   Set Comps = Document.ComponentSet

      ' Make sure the Query component exists
   If Comps.ItemByName("Query") < 0 Then
       Set Qry = Document.NewQuery("Query")
   Else
       Set Qry = Comps.Item("Query")
   End If

   Qry.Text = "Select Top 1 cCoordSys([" & Geom_ColumnName & "]) as [CS] FROM [" &
SourceGeom_TableName & "];"
   Qry.RunEX TRUE
   Set Tbl = Qry.Table
   Set Rcrds = Tbl.RecordSet

   Set Extract_CoordSys_From_Geom = _
         Application.NewCoordinateSystemFromBinary(Rcrds(0).Data("CS"))

End Function

' ********************************************************************

Private Function Table_Column_Exists( _
        ByVal Table_Name, _
        ByVal Column_Name, _
        ByVal Column_Type)

      ' Author: Lorne Ketch
      ' Email:  LKetch|at|hfx[1st_dot]eastlink{2nd_dot}ca

      ' Column_Type is a ColumnType enumeration.
```

```
   ' ColumnTypeAChar       ColumnTypeGeom       ColumnTypeInt8U
   ' ColumnTypeAText       ColumnTypeGeomSHP    ColumnTypeLatitude
   ' ColumnTypeBinary      ColumnTypeGeomWKB    ColumnTypeLongitude
   ' ColumnTypeBoolean     ColumnTypeInt16      ColumnTypePercentage
   ' ColumnTypeCoordSys    ColumnTypeInt16U     ColumnTypeTime
   ' ColumnTypeCurrency    ColumnTypeInt32      ColumnTypeURL
   ' ColumnTypeFloat32     ColumnTypeInt32U     ColumnTypeWChar
   ' ColumnTypeFloat64     ColumnTypeInt8       ColumnTypeWText

Dim Comps
Dim Source_Tbl, Source_Cols, Source_Col

Table_Column_Exists = TRUE

Set Comps = Document.ComponentSet

    ' Does the Component exist?
If Comps.ItemByName(Table_Name) < 0 Then
    Application.MessageBoxEX _
        "Component """ & Table_Name & """ does not exist in the project.", _
        "Routine: Table_Column_Exists", _
         MessageBoxTypeIconError OR MessageBoxTypeOk
    Table_Column_Exists = False
    Exit Function
End If

Set Source_Tbl  = Comps.Item(Table_Name)

    ' Is the Component a Table?
If Source_Tbl.Type <> ComponentTable Then
    Application.MessageBoxEX _
        "Component """ & Table_Name & """ is not a Table.", _
        "Routine: Table_Column_Exists", _
         MessageBoxTypeIconError OR MessageBoxTypeOk
    Table_Column_Exists = False
    Exit Function

End If

Set Source_Cols = Source_Tbl.ColumnSet

    ' Does the Column exist?
On Error Resume Next
Set Source_Col  = Source_Cols.Item(Column_Name)
If Err Then
   Application.MessageBoxEX _
       "Table:  " & Table_Name  & VbCrLf & VbCrLf & _
       "Column """ & Column_Name & """ does not exist in the table.", _
       "Routine: Table_Column_Exists", _
        MessageBoxTypeIconError OR MessageBoxTypeOk
   Err.Clear
   On Error GoTo 0
   Table_Column_Exists = False
   Exit Function
 End If

If Source_Col.Type <> Column_Type Then
```

```
      Application.MessageBoxEX _
          "Table:     " & VbTab & Table_Name  & VbCrLf & _
          "Column:    " & VbTab & Column_Name & VbCrLf & _
          "Column Type:" & VbTab & Source_Col.TypeName & VbCrLf & VbCrLf & _
          "ERROR: Incorrect Column Type" , _
          "Routine: Table_Column_Exists", _
           MessageBoxTypeIconError OR MessageBoxTypeOk
      Table_Column_Exists = False
      Exit Function
   End If

End Function

' ****************************************************************************
```

XML – Custom Styles, etc

Defining Custom Label Styles (Points and Areas can be done as well)

Manifold©-L
Author: Adam Wachowski
Date: 01-13-2006 08:18
http://lists.directionsmag.com/discussion/read.php?f=29&i=41329&t=41325

```
> I would like to be able to edit a style that draws
> labels in a box. I would like to control the width
> of the box because building a map with a high
> density of labels. Is there any way of doing this?

Yes, with a custom label style, eg:

<xml>
  <style>
    <name>MyLabel</name>
    <type>label</type>
    <code>
      <rectangle xmin="-2000" ymin="-1000" xmax="2000" ymax="1000" />
      <text style="plain" />
    </code>
  </style>
</xml>

--
Adam Wachowski
Manifold Development Team
```

See: "Custom Scalable Styles for Points, Areas and Labels" help section (LAK)

Following custum lines style taken from:
KlausDE on 11/30/2006 8:22 AM (#31175)
http://forum.manifold.net/Site/Thread.aspx?id=31139

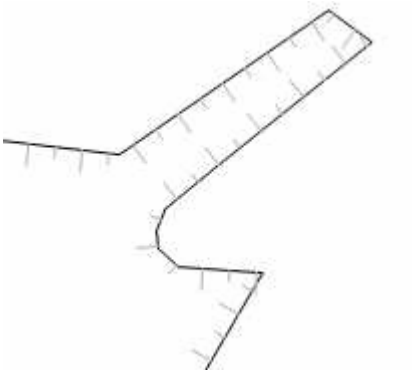
Line: River Bank

File Name: SlopeStyle.xml

```
<xml>
  <style>
    <name>Bank</name>
    <type>line</type>
    <preview>36</preview>
    <code>
      <refLine size="20" style="solid" />
      <polygon fore="b">
        <point x="123" y="0" />
        <point x="123" y="100" />
        <point x="127" y="100" />
        <point x="127" y="0" />
      </polygon>
      <polygon fore="b">
        <point x="373" y="0" />
        <point x="373" y="200" />
        <point x="377" y="200" />
        <point x="377" y="0" />
      </polygon>
      <polygon fore="b">
        <point x="623" y="0" />
        <point x="623" y="100" />
        <point x="627" y="100" />
        <point x="627" y="0" />
      </polygon>
      <polygon fore="b">
        <point x="873" y="0" />
        <point x="873" y="200" />
        <point x="877" y="200" />
        <point x="877" y="0" />
      </polygon>
    </code>
  </style>
</xml>
```

KlausDE on 11/30/2006 8:46 AM (#31212)

Copy SlopeStyle.xml into the config folder and start a new Manifold session. You need to set the size to pretty high values.



Point: Hourglass

seatrails on 1/9/2007 7:41 AM (#33216)

http://forum.manifold.net/Site/Thread.aspx?id=33191

Save the following into a .xml file in the Manifold Program Config directory:

```
<xml>
<style>
<name>hourglass</name>
<type>point</type>

  <polygon>
   <point x="500" y="-1000" />
   <point x="400" y="-900" />
   <point x="500" y="-800" />
   <point x="400" y="-800" />
   <point x="400" y="-600" />
   <point x="300" y="-300" />
   <point x="100" y="-100" />
   <point x="0" y="0" />
   <point x="100" y="100" />
   <point x="300" y="300" />
   <point x="400" y="600" />
   <point x="400" y="800" />
   <point x="500" y="800" />
   <point x="400" y="900" />
   <point x="500" y="1000" />
   <point x="-500" y="1000" />
   <point x="-400" y="900" />
   <point x="-500" y="800" />
   <point x="-400" y="800" />
   <point x="-400" y="600" />
   <point x="-300" y="300" />
   <point x="-100" y="100" />
   <point x="0" y="0" />
   <point x="-100" y="-100" />
   <point x="-300" y="-300" />
   <point x="-400" y="-600" />
   <point x="-400" y="-800" />
   <point x="-500" y="-800" />
   <point x="-400" y="-900" />
   <point x="-500" y="-1000" />
  </polygon>

</style>
</xml>
```

Example Equal-Interval Theme

http://69.17.46.171/Site/Thread.aspx?id=22795&ti=632901092226900000

rheitzman at 6/21/2006 2:50 PM (#23054)

Something to watch for using this method is that the resulting Equal Count distribution is limited to the data in the sample which may not reflect a typical range of data. Equal Interval may be a better stat, but it still has data set related issues. One way to deal with this is to make a Theme.xml that is close then hand tweak it to cover the expected values. (The theme from either Interval type is identical.) For example:

```
<xml>
  <theme>
    <type>color[/type>
    <interval>true[/interval>
    <column>Globvalue[/column>
    <colors>
      [color>#f5e60c[/color>
      [color>#65e13d[/color>
      [color>#00b23d[/color>
      [color>#067d1d[/color>
      [color>#1f4791[/color>
    </colors>
    <colorMin>#cd853f[/colorMin>
    <colorMax>#1f4791[/colorMax>
    <colorDef>#cd853f[/colorDef>
    <values>
      [value>0.00[/value>
      [value>1.00[/value>
      [value>2.00[/value>
      [value>3.00[/value>
      [value>4.00[/value>
    </values>
  </theme>
</xml>
```

Adding a True Type Font Symbol

http://69.17.46.171/Site/Thread.aspx?id=19413&ti=632774325922900000

nujseyer on 3/6/2006 4:38 PM (#19422)

```
<xml>
  <style>
    <name>ESRI Cartography #00F6</name>
    ESRI Cartography
    <symbol>#00F6</symbol>
    <type>point</type>
  </style>
    <name>ESRI Cartography #002C</name>
    ESRI Cartography
    <symbol>#002C</symbol>
    <type>point</type>
  </style>
</xml>
```

http://forum.manifold.net/Site/Thread.aspx?id=31629&ti=633010714970600000

c.calhoun on 12/7/2006 5:45 AM (#31632)

```
<xml>
  <style>
    <name>ESRI Environmental&Icons #33</name>
    ESRI Environmental&Icons
    <symbol>#33</symbol>
    <type>point</type>
  </style>
</xml>
```

c.calhoun on 12/7/2006 5:47 AM (#31635)

Apparently it just reformats it here, SO just put '&' (without the quotes) in replace of your & in the <name> and tags and it should work

c.calhoun on 12/7/2006 5:48 AM (#31636)

really...last time here. I don't know why this won't let me post the text properly

put '& amp ;' - without any quotes and without any spaces.

XML Schema CANNOT be used to validate a File

Cannot retrieve the URL:
Many June and July 2006 postings seem to be lost at:
http://lists.directionsmag.com/discussion/list.php?f=29
and
http://lists.directionsmag.com/discussion/search.php?f=29

From: adamw@manifold.net
To: manifold-l@lists.directionsmag.com
Sent: Tue 06/27/2006 10:00 AM
Subject: RE: [Manifold-l] GPX support

First off, I agree that it would be good to be able to import GPX and KML files into Manifold. If enough people request this functionality, I have no doubts it will appear in future updates.

With that taken care of, I would like to address what seems to be a common misconception with regard to the use of XML as an interchange format:

> Scisoft said it all: a GPX file (or any XML format) should be read using the official DTD,
> XML Schema or DSD technology that is published and purports to describe the semantic
> structure of the XML data.

Having a schema for an XML file is not nearly enough to be able to read data within that file and make sense of it. A schema can be used to *validate* the file, that is, given an XML file and a schema, it can be easily determined whether or not the XML file conforms to the schema.

However, the application reading the XML file still has to parse individual tags within the file and somehow organize the resulting information, even if it knows that the XML file conforms to a particular schema.

Let me provide an example.

Here is an XML:

```
<?xml version="1.0" encoding="utf-8"?>
<x:a
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:x="urn:data"
  xsi:schemaLocation="urn:data http://server/data.xsd">
  <b>15.5</b>
  <c>-28.0</c>
  <d>3.0</d>
</x:a>
```

Here is an XML schema:

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema targetNamespace="urn:data"
  xmlns="http://tempuri.org/XMLSchema.xsd"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="a">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="b" type="xs:float" />
        <xs:element name="c" type="xs:float" />
        <xs:element name="d" type="xs:float" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The XML conforms to the schema (you can take my word for it). Now, armed with that knowledge, can you tell me what is the meaning of the numeric values inside the XML? How do you want Manifold to import these values? Should Manifold create a table with three columns containing the values of 15.5, -28.0 and 3.0? Or should it create a drawing with a point at 15.5, -28.0 and a column value of 3.0? Or should it create a drawing with a circle centered at 15.5, -28.0 with a radius of 3.0? Of course, there is no way to tell. The schema does not contain this kind of information.

An XML schema does *not* describe the semantics of the XML subformat it validates. Having a schema is not a substitute for having the format documentation. While it is possible to write code that would read an XML file without any prior knowledge about the contents of that file except that it conforms to some schema (which is determined dynamically in the process of importing the file or is not backed up by the format documentation), this code will be much too generic to be useful. It would be easy to import a GPX file as a comments component, but this is not what you want, right? :-)

Adam Wachowski

Rotating Symbols on a Continuous Scale

http://forum.manifold.net/Site/Thread.aspx?id=30442&ti=632995763476730000

KlausDE on 11/19/2006 11:39 PM (#30526)

Seems high time for rephrasing of "continuous shading" in the format dialog as the continuous values are not restricted to colors. Nick Verge suggested 'continuous variation' or perhaps one could have only 'continuous' there - it's all kind of values of course.

Copy the following lines into an XML-file, adapt the name of the column to your rotation data and load it in the thematic formatting dialog. Then manually check 'continuous shading' (it's not saved with the theme!!! Why not???) and you're prepared for all angles that might come.

```
<xml>
  <theme>
    <type>angle</type>
    <interval>true</interval>
    <column>Angle</column>
    <angles>
      <angle>0</angle>
      <angle>360</angle>
    </angles>
    <angleMin>0</angleMin>
    <angleMax>360</angleMax>
    <angleDef>0</angleDef>
    <values>
      <value>0.00</value>
      <value>360.00</value>
    </values>
  </theme>
</xml>
```

Custom Layout Templates

http://69.17.46.171/Site/Thread.aspx?id=26722&ti=632928851816070000

kgf at 8/30/2006 1:28 PM (#26723)

Hi all. Can anyone help with custom templates?. I have layout templates working fine either from Apply File or from adding to the Config folder. However the problem I have is that they seem to be bound to the name of the Map they were created from. So if I create a template using Map1 and then apply a template to a layout attached to Map2 the template uses Map1. The workaround is to create a layout using default name Map each time, but is there a way to edit the

xml file to make the layout refer to the active Map rather than the name of the Map the template was created with?.

Thanks

adamw at 9/2/2006 6:20 AM (#26879)
Use layouts bound to components and use < body ... > instead of < component name=... >.

kgf on 9/3/2006 12:59 PM (#26957)
Thanks very much, I changed

```
<component background="auto" border="thin" ....
```
to
```
<body background="auto" border="thin" .....
```

and all seems to work now.

VB.NET Reader for XML Units File

David M Brubacher OLS OLIP
11/24/2003 3:55 AM
http://www.georeference.org/Forums/tabid/71/forumid/3/tpage/1/view/topic/postid/124/Default.aspx#180

Here is a small console application that reads the above XML and dumps each unit to screen:

Module ListUnits

```
  ' Application start point
Sub Main()
   Dim reader As XmlReader

   Try
        ' <- change the path to suit your system
      reader = New XmlTextReader("c:\documents\units.xml")

      Dim unitName As String
      Dim unitScale As Double

      ' read through XML file
      While reader.Read()

           ' start reading new unit
        If reader.NodeType = XmlNodeType.Element And reader.Name = "unit" Then
          unitName = ""
          unitScale = 0

            ' accept unit name
        ElseIf reader.NodeType = XmlNodeType.Element And reader.Name = "name" Then
          unitName = reader.ReadString()
```

```
            ' accept unit scale
          ElseIf reader.NodeType = XmlNodeType.Element And reader.Name = "scale" Then
            unitScale = Double.Parse(reader.ReadString())

            ' finish reading unit
          ElseIf reader.NodeType = XmlNodeType.EndElement And reader.Name = "unit" Then
            Console.WriteLine("{0}: {1}", unitName, unitScale)
          End If
        End While

        Finally
        reader.Close()
    End Try
End Sub
```